

# A STUDY OF THE STRUCTURES OF BOOLEAN FUNCTIONS AND ITS APPLICATION TO THE SYNTHESIS OF SWITCHING CIRCUITS

ICHIZO NINOMIYA

*Department of Applied Physics*

(Received December 31, 1961)

## Preface

The Boolean algebra was devised already in the middle of the nineteenth century by an Englishman George Boole as an algebra of calculus of propositions. But it is as late as in 1938 that the algebra was put to use in the study of switching circuits by C. E. Shannon. Thenceforth, the analysis and the synthesis of switching circuits which had been carried out only by skill and experiences have largely been systematized. This epoch-making progress in the study of switching circuits has caused, in conjunction with the progress of electronics, today's brilliant development in electronic digital computers.

Although the application of the Boolean algebra to the study of switching circuits has achieved a satisfactory result, this does not mean that there are no problems left in this field. In reality, there remain a number of unsolved or unexplored problems which call for creative activities of mathematicians and engineers.

One of such problems is concerned with the search of a general method for synthesizing the simplest switching circuit under a given standard of simplicity. A simple circuit is desirable not only by simplicity but also by easy maintenance and by high reliability. Further, although the profit gained by the use of simple circuits may not be important in a single circuit, the total effect will be considerable when the same circuit is used hundreds of times in a huge apparatus like a digital computer. In this sense, the solution of the above mentioned problem has a primary importance. In spite of its importance, however, the solution is very difficult, and it has not been so far attained even partially.

As a worker in this field, the author holds a conviction that the clue of this and other similar problems, if any, must be a penetrating insight into the structure of Boolean functions representing the operations of switching circuits. Under the structure of Boolean functions, we mean vaguely some intrinsic properties such as symmetry or functional separability. It cannot be defined precisely, because what is the structure is not an established proposition but a crude notion to be clarified hereafter.

The present paper is the report of the work carried out by the author with the above object. The contents will now be explained chapter by chapter.

The first two chapters are written as the preliminaries for the later chapters. In Chapter 1, the fundamental concepts of the general Boolean algebras are ex-

plained. Especially, the lattice theoretical and ring theoretical background of Boolean algebras are clarified. In Chapter 2, the fundamental theories of Boolean functions are described systematically together with the author's contributions.

In Chapter 3, a new theory concerning the coordinate representation of Boolean functions are developed. It is shown that the coordinate representation permits a quick and easy recognition of various structures and leads to the discovery of a new structure, by which new concepts for the classification of Boolean functions are introduced.

Chapter 4 deals with the classification of Boolean functions of four variables as an application of the theory of Chapter 3. Any two Boolean functions are said to be of the same type, if and only if one of them can be transformed into the other by a symmetry (a permutation and/or a complementation of variables). Under the classification we mean to classify Boolean functions into types and to select a typical function from each type. Functions of the same type are very alike in many respects and the switching circuits realizing them are physically similar. Herein consists the significance of the classification. However, on account of the prohibitive largeness of the number of types, it is impracticable for the case of more than four variables. The classification of Boolean functions of four variables has been carried out in the Computation Laboratory of Harvard University, and the result has been published in the Appendix to the book, "Synthesis of Electronic Computing and Control Circuits". But the method used was primitive and the accomplishment of the classification was possible only by the help of a computer. The author reattacked this problem on the basis of a new principle utilizing the knowledge of new structures of Boolean functions obtained in Chapter 3. The process of the classification is described in detail in Section 4.4. It is shown there that the process is largely simplified by virtue of the new principle. The result is tabulated in Appendix 1. In the table, numerous informations concerning the structures of Boolean functions which are not contained in the Harvard Table are added newly.

In order to make the significance of the table more thorough, the author synthesized switching circuits for all types of Boolean functions of four variables which are minimal under rational standards of simplicity. The results are collected in the table of Appendix 2. The table contains relay circuits, rectifier circuits, vacuum tube circuits and transistor circuits. The summarized properties of minimal switching circuits for Boolean functions of four variables and the comparisons of the table with the Harvard Table (on vacuum tube circuits) and Moore's table (on relay circuits) are stated in Chapter 7.

In Chapter 5 and Chapter 6, various existing methods for the synthesis of relay switching circuits (Chapter 5) and electronic switching circuits (Chapter 6) are explained and supplemented with some new methods devised by the author himself. These methods are illustrated by numerous examples based on the experiences of the author. The examples are mainly concerned with switching circuits for functions of four variables, but the methods are also useful for the synthesis of switching circuits for functions of moderate number of variables. In these Chapters, the emphasis is laid on the importance of the direct grasp of the structures of Boolean functions.

In retrospect, a certain progress has been made in the application of Boolean algebra to the study of switching circuits. However, many important problems

are still left unsolved. The author is anxious to struggle with these problem also in future.

The author wishes to thank professor Kazuo Kondo of Tokyo University and Professor Zyurō Sakadi of Nagoya University for their valuable criticisms and continual encouragements.

## CONTENTS

1. Theory of Boolean Algebras.....	152
1.1. Postulates of Boolean Algebras.....	152
1.2. Fundamental Theorems .....	154
1.3. Boolean Algebras as Lattices.....	157
1.4. Boolean Algebras as Rings.....	158
1.5. Examples of Boolean Algebras .....	160
1.5.1. Binary Functions.....	160
1.5.2. Set Calculus.....	160
1.5.3. Calculus of Propositions.....	161
2. Theory of Boolean Functions.....	162
2.1. Boolean Functions.....	162
2.2. Linear Functions.....	166
2.3. Automorphisms and Symmetries of $B_n$ .....	168
2.4. Functionally Separable Boolean Functions.....	171
2.5. Monotonous Boolean Functions.....	176
2.6. Geometric Representations of Boolean Functions .....	178
2.6.1. Representation by $n$ -Cube .....	178
2.6.2. Representation by $2^n$ -Cube.....	179
2.6.3. Representation by the Karnaugh Map .....	179
2.7. Simplification of Boolean Functions .....	183
2.7.1. Quine's Algorithm.....	183
2.7.2. Simplification by the Karnaugh Map .....	192
3. Theory of Coordinate Representation of Boolean Functions .....	194
3.1. Boolean Matrix Representation of Boolean Functions.....	194
3.2. Real Matrix Representation of Boolean Functions.....	195
3.3. Coordinate Representation of Boolean Functions.....	196
3.4. Some Useful Theorems.....	201
3.5. Coordinate Representation of Symmetry Structures .....	206
3.6. Coordinate Representation of Functional Separability.....	210
3.7. Divisible Boolean Functions .....	213
3.8. Linear Automorphisms and Linear Transformations.....	215
3.9. Geometric Derivation of Coordinate Representation .....	217
4. Classification of Boolean Functions of Three and Four Variables .....	219
4.1. Significance of Classification of Boolean Functions .....	219
4.2. Principle of the Method of Classification.....	221
4.3. Classification of Boolean Functions of Three Variables.....	223
4.3.1. Classification of Functions with the Index 4.....	223
4.3.2. Classification of Functions with the Index 3.....	224
4.3.3. Classification of Functions with the Index 2.....	224
4.3.4. Conclusion.....	226
4.4. Classification of Boolean functions of Four Variables.....	227
4.4.1. Classification of Functions with the Index 8.....	227
4.4.2. Classification of Functions with the Index 7.....	228
4.4.3. Classification of Functions with the Index 6.....	228
4.4.4. Classification of Functions with the Index 5.....	229

4.4.5. Classification of Functions with the Index 4.....	231
4.4.6. Classification of Functions with the Index 3.....	234
4.4.7. Classification of Functions with the Index 2.....	236
4.4.8. Conclusion.....	237
4.4.9. Comparison of the New Table with the Harvard Table.....	240
5. Synthesis of Relay Switching Circuits.....	241
5.1. Relay Switching Circuits.....	241
5.2. Expansion Method.....	244
5.2.1. Disjunctive Tree Method.....	244
5.2.2. Sandwich Method.....	248
5.2.3. Cross Connection Method.....	249
5.2.4. Square Connection Method.....	250
5.3. Absorption Method.....	252
5.4. Path Accumulation Method.....	253
5.5. Double Complementation Method.....	259
5.6. Synthesis of Circuits for Symmetric Functions.....	262
5.7. Boolean Matrix Method.....	265
6. Synthesis of Electronic Switching Circuits.....	270
6.1. Electronic Switching Circuits.....	270
6.2. Rectifier Switching Circuits.....	271
6.3. Vacuum tube Switching Circuits.....	275
6.4. Transistor Switching Circuits.....	284
7. Minimal Switching Circuits for Boolean Functions of Four Variables.....	292
7.1. Minimal Relay Switching Circuits.....	293
7.2. Minimal Rectifier Switching Circuits.....	293
7.3. Minimal Vacuum Tube Switching Circuits.....	294
7.4. Minimal Transistor Switching Circuits.....	294
Appendix 1. Table of Boolean Functions of Four Variables.....	295
Appendix 2. Table of Minimal Switching Circuits for Boolean Functions of Four Variables.....	301
References.....	362

## 1. Theory of Boolean Algebras

### 1.1. Postulates of Boolean Algebras

The *Boolean algebra* is a class  $B$  with two rules of combination satisfying a certain system of postulates. There are many equivalent systems of postulates each of which can be used to define a Boolean algebra. From among such systems, we here adopt the following one known as Huntington's<sup>1)</sup>.

*Postulate 1 a.* If  $x \in B$  and  $y \in B$ , then  $x + y \in B$ .

*Postulate 1 b.* If  $x \in B$  and  $y \in B$ , then  $x \cdot y \in B$ .

*Postulate 2 a.* There is an element  $0 \in B$  such that  $x + 0 = x$  for every  $x \in B$ ,

*Postulate 2 b.* There is an element  $1 \in B$  such that  $x \cdot 1 = x$  for every  $x \in B$ .

*Postulate 3 a.* For any  $x$  and  $y$  in  $B$ ,  $x + y = y + x$ .

*Postulate 3 b.* For any  $x$  and  $y$  in  $B$ ,  $x \cdot y = y \cdot x$ .

*Postulate 4 a.* For any  $x$ ,  $y$  and  $z$  in  $B$ ,  $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$ .

*Postulate 4 b.* For any  $x$ ,  $y$  and  $z$  in  $B$ ,  $x + (y \cdot z) = (x + y) \cdot (x + z)$ .

*Postulate 5.* If the elements  $0$  and  $1$  of Postulates 2 are unique, then, for

<sup>1)</sup> cf. Ref. 27, p. 31.



every  $x \in B$ , there is an element  $x' \in B$  such that  $x + x' = 1$  and  $x \cdot x' = 0$ .

*Postulate 6.* There are at least two distinct elements in  $B$ .

The first eight postulates are grouped together in four sets of each two, and this grouping illustrates the perfect symmetry or *duality* of the algebra with respect to the operations “+” and “·”.<sup>1)</sup> In fact, if, in any of these postulates, 0 is replaced by 1, 1 by 0, each + by ·, and each · by +, the result will be the *dual* of the original postulate. This duality will appear also when we begin to derive theorems from the postulates, for, if we are given some theorem, we can immediately construct a proof of dual theorem starting with the dual expressions and justifying each step by the dual postulates.

The two operations are called the *addition* and the *multiplication*. The symbol “·” for the multiplication will be omitted whenever this does not lead to confusion. Thus, for example, the expression  $x + (y \cdot z)$  will be written as  $x + yz$ .

Postulates 3 are called the *commutative laws* and Postulates 4 are called the *distributive laws*. The *associative laws* are not postulated. They will be derived from the postulates as shown in the next section.

Now, for the sake of illustration, we consider a class  $B_0$  of exactly two elements, 0 and 1, where the two operations are defined by:

$$\begin{aligned} 0+0=0, \quad 0+1=1+0=1, \quad 1+1=1, \\ 1 \cdot 1=1, \quad 1 \cdot 0=0 \cdot 1=0, \quad 0 \cdot 0=0. \end{aligned}$$

Let us now prove that  $B_0$  satisfies the system of postulates and hence is a Boolean algebra.

*Postulate 1 a:* Evident.

*Postulate 1 b:* Evident.

*Postulate 2 a:* Since  $0+0=0$  and  $1+0=1$ , 0 is really a 0-element.

*Postulate 2 b:* Since  $1 \cdot 1=1$  and  $0 \cdot 1=0$ , 1 is really a 1-element.

*Postulate 3 a:* Evident.

*Postulate 3 b:* Evident.

*Postulate 4 a:* Examining all the eight cases, we obtain the results shown in Table 1.1.1, whose fifth column and eighth column indicate that the postulate is satisfied.

*Postulate 4 b:* Examining all the eight cases, we obtain the results shown in Table 1.1.1, whose tenth column and thirteenth column indicate that the postulate is satisfied.

*Postulate 5:*  $0+1=1+0=1$  and  $0 \cdot 1=1 \cdot 0=0$ . Hence, if we put  $0'=1$  and

<sup>1)</sup> Any two symbols, e.g.,  $\vee$  and  $\wedge$ , may be used for these operations. Here we follow the latest engineering trend.

$1' = 0$ , the postulate is satisfied.

*Postulate 6:* Evident.

TABLE 1.1.1

$x$	$y$	$z$	$y+z$	$x(y+z)$	$xy$	$xz$	$xy+xz$	$yz$	$x+yz$	$x+y$	$x+z$	$(x+y)(x+z)$
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0	1	0
0	1	0	1	0	0	0	0	0	0	1	0	0
0	1	1	1	0	0	0	0	1	1	1	1	1
1	0	0	0	0	0	0	0	0	1	1	1	1
1	0	1	1	1	0	1	1	0	1	1	1	1
1	1	0	1	1	1	0	1	0	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1

This simple Boolean algebra  $B_0$  is called the *two-element Boolean algebra*<sup>1)</sup>. It is observed that  $B_0$  is isomorphic with a field with the character 2.

## 1.2. Fundamental Theorems

A series of theorems will now be derived from the postulates. The theorems, like the postulates, will be presented in pairs where each theorem is the dual of the one paired with it.

*Theorem 1.2.1 a.* The element 0 of Postulate 2 a is unique.

*Proof.* Suppose there be two 0-elements called  $0_1$  and  $0_2$ . Then, for every element  $x$ , we have:  $x+0_1=x$  and  $x+0_2=x$ . Now substitute  $x=0_2$  in the first equation and  $x=0_1$  in the second equation. Then we obtain:  $0_2+0_1=0_2$  and  $0_1+0_2=0_1$ . But  $0_2+0_1=0_1+0_2$ . Therefore  $0_2=0_1$ .

*Theorem 1.2.1 b.* The element 1 of Postulate 2 b is unique.

*Theorem 1.2.2 a.* For every  $x$ ,  $x+x=x$ .

*Proof.*  $x+x=(x+x) \cdot 1=(x+x)(x+x')=x+xx'=x+0=x$ .

*Theorem 1.2.2 b.* For every  $x$ ,  $xx=x$ .

*Theorem 1.2.3 a.* For every  $x$ ,  $x+1=1$ .

*Proof.*  $x+1=(x+1) \cdot 1=(x+1)(x+x')=x+1 \cdot x'=x+x'=1$ .

*Theorem 1.2.3 b.* For every  $x$ ,  $x \cdot 0=0$ .

*Theorem 1.2.4 a.* For every  $x$  and  $y$ ,  $x+xy=x$ .

*Proof.*  $x+xy=x \cdot 1+xy=x(1+y)=x \cdot 1=x$ .

*Theorem 1.2.4 b.* For every  $x$  and  $y$ ,  $x(x+y)=x$ .

<sup>1)</sup> This is the special case for  $n=0$  of  $B_n$ . See page 11,

Theorems 1.2.4 are called the *absorptive laws*.

*Theorem 1.2.5. For every  $x$ ,  $x'$  is unique.*

*Proof.* Suppose  $x$  has  $x'_1$  and  $x'_2$  such that  $x + x'_1 = x + x'_2 = 1$  and  $xx'_1 = xx'_2 = 0$ . Then  $x'_2 = 1 \cdot x'_2 = (x + x'_1)x'_2 = xx'_2 + x'_1x'_2 = 0 + x'_1x'_2 = x'_1x + x'_1x'_2 = x'_1(x + x'_2) = x'_1 \cdot 1 = x'_1$ . Hence there is only one  $x'$  and this called the *complement* of  $x$ . For example, the elements 0 and 1 are the complement of one another. In view of the uniqueness of the complement, the unary operation  $x \rightarrow x'$  can be defined in Boolean algebras. The operation is called the *complementation*.

*Theorem 1.2.6. For every  $x$ ,  $(x')' = x$ .*

*Proof.* Since  $x' + x = 1$  and  $x'x = 0$ ,  $x$  is the complement of  $x'$ , i.e.,  $(x')' = x$ .

*Theorem 1.2.7 a. For every  $x$  and  $y$ ,  $(x+y)' = x'y'$ .*

*Proof.* We shall prove that  $(x+y) + x'y' = 1$  and  $(x+y) \cdot x'y' = 0$ . Then it is clear that  $x+y$  and  $x'y'$  are the complement of one another. In order to carry out the above plan, we need the following two lemmas.

*Lemma 1.2.1 a. For every  $x$  and  $y$ ,  $x + (x' + y) = 1$ .*

*Proof.*  $x + (x' + y) = 1 \cdot [x + (x' + y)] = (x + x')[x + (x' + y)]$   
 $= x + x'(x' + y) = x + x' = 1$ .

*Lemma 1.2.1 b. For every  $x$  and  $y$ ,  $x(x'y) = 0$ .*

We are now ready to proceed with the plan.

$$\begin{aligned} (x+y) + x'y' &= [(x+y) + x'][(x+y) + y'] \\ &= [x' + [(x')' + y]][y' + [(y')' + x]] = 1 \cdot 1 = 1 \\ (x+y) \cdot x'y' &= x(x'y') + y(x'y') = x(x'y') + y(y'x') = 0 + 0 = 0. \end{aligned}$$

This completes the proof.

*Theorem 1.2.7 b. For every  $x$  and  $y$ ,  $(xy)' = x' + y'$ .*

Theorems 1.2.7 are called *De Morgan's laws* and are very useful in manipulating Boolean expressions.

*Example 1.2.1. Find the complement of  $u[v' + (wx' + y'x)]$ .*

*Solution.*  $[u[v' + (wx' + y'x)]]' = u' + [v' + (wx' + y'x)]' = u' + v(w'x' + y'z)'$   
 $= u' + v[(wx')'(y'z)] = u' + v[(w' + x)(y + z')]$ .

From the results of this example, it will be seen that De Morgan's laws can be generalized to the following rule: *To find the complement of a Boolean expression, replace each  $+$  by  $\cdot$ , each  $\cdot$  by  $+$  and each letter by its complement.*

*Theorem 1.2.8 a. For every  $x$ ,  $y$  and  $z$ ,  $(x+y) + z = x + (y+z)$ .*

*Proof.* The proof of this important theorem is somewhat difficult. Hunting-

ton's plan for the proof is as follows: Let  $(x+y)+z=u$  and  $x+(y+z)=v$ . Then show that  $v+u'=1$  and  $vu'=0$ . If these relations are true,  $u'$  and  $v$  are the complement of one another, and consequently,  $(u')'=u=v$ .

$$\text{Now } v+u' = v + (x'y')z' = (v+x'y')(v+z') = [(v+x')(v+y')](v+z').$$

Here the three expressions  $v+x'$ ,  $v+y'$  and  $v+z'$  are evaluated as follows.

$$\begin{aligned} v+x' &= x' + [x + (y+z)] = x' + [(x')' + (y+z)] = 1. \\ v+y' &= (y' + y)(y' + v) = y' + yv = y' + y[x + (y+z)] = y' + [xy + y(y+z)] \\ &= y' + (xy + y) = y' + y = 1. \\ v+z' &= (z' + z)(z' + v) = z' + zv = z' + z[x + (y+z)] = z' + [xz + z(y+z)] \\ &= z' + (xz + z) = z' + z = 1. \end{aligned}$$

$$\text{Therefore } v+u' = [(v+x')(v+y')](v+z') = (1 \cdot 1) \cdot 1 = 1 \cdot 1 = 1.$$

$$\text{Again } vu' = u'[x + (y+z)] = u'x + u'(y+z) = u'x + (u'y + u'z).$$

The three expressions  $u'x$ ,  $u'y$  and  $u'z$  are evaluated as follows.

$$\begin{aligned} u'x &= xx' + xu' = x(x' + u') = x[x' + (x'y')z'] = x[(x' + x'y')(x' + z')] \\ &= x[x'(x' + z')] = xx' = 0. \\ u'y &= yy' + yu' = y(y' + u') = y[y' + (x'y')z'] = y[(y' + x'y')(y' + z')] \\ &= y[y'(y' + z')] = yy' = 0. \\ u'z &= z[z'(x'y')] = 0. \end{aligned}$$

Therefore  $vu' = u'x + (u'y + u'z) = 0 + (0 + 0) = 0 + 0 = 0$ . This completes the proof.

*Theorem 1.2.8 b. For every  $x, y$  and  $z$ ,  $(xy)z = x(yz)$ .*

Theorems 1.2.8 are the *associative laws*. By virtue of the laws, we can write Boolean sums and products without parentheses just as in ordinary algebra. Thus, for example, the solution to Example 1.2.1 can be written as:  $[u(v' + wx' + y'z)]' = u' + v(w' + x)(y + z')$ .

*Theorem 1.2.9 a. For every  $x$  and  $y$ ,  $x + x'y = x + y$ .*

*Proof.*  $x + x'y = (x + x')(x + y) = 1 \cdot (x + y) = x + y$ .

*Theorem 1.2.9 b. For every  $x$  and  $y$ ,  $x(x' + y) = xy$ .*

*Theorem 1.2.10. For every  $x, y$  and  $z$ ,  $(x + y)(x' + z) = x'y + xz$ .*

$$\begin{aligned} \text{Proof. } (x + y)(x' + z) &= xx' + x'y + xz + yz = 0 + x'y + xz + yz \cdot 1 \\ &= x'y + xz + yz(x + x') = x'y + xz + x'yz + xyz \\ &= x'y(1 + z) + xz(1 + y) = x'y \cdot 1 + xz \cdot 1 \\ &= x'y + xz. \end{aligned}$$

Note that Theorem 1.2.10 is equivalent to its dual:  $xy + x'z = (x' + y)(x + z)$ .

*Theorem 1.2.11 a. For every  $x, y$  and  $z$ ,  $[(x + y)(x' + z)]' = (x + y')(x' + z')$ .*

*Proof.*  $[(x+y)(x'+z)]' = (x'y+xz)' = (x+y')(x'+z')$ .

*Theorem 1.2.11 b.* For every  $x, y$  and  $z$ ,  $(xy+x'z)' = xy' + x'z'$ .

### 1.3. Boolean Algebras as Lattices

For any two elements,  $x$  and  $y$ , of a Boolean algebra such that  $x+y=y$ , we write  $x \leq y$  or  $y \geq x$  and read " $y$  includes  $x$ ". The relation of inclusion, thus defined, is by no means indispensable to the theory of Boolean algebras, but it is frequently convenient as well as elegant to use this concept. In this section, a series of theorems concerning the relation of inclusion will be derived.

*Theorem 1.3.1.* The relation  $x \leq y$  is equivalent to any of the following four relations: (1)  $x+y=y$ , (2)  $xy=x$ , (3)  $x'+y=1$ , (4)  $xy'=0$ .

*Proof.* We shall prove the theorem by showing that (1) implies (3), (3) implies (4), (4) implies (2) and (2) implies (1).

Assume (1), then  $x'+y = x' + x + y = 1 + y = 1$ . It follows that (1) implies (3). Next, assume (3), then  $xy' = (x' + y')' = 1' = 0$ . It follows that (3) implies (4). Further assume (4), then  $xy = xy + xy' = x(y + y') = x \cdot 1 = x$ . It follows that (4) implies (2). Assume, at last, (2), then  $x + y = xy + y = y$ . It follows that (2) implies (1). This completes the proof.

Note that  $x \leq y$  and  $x \geq y$  are the dual of one another, since they are equivalent to  $x+y=y$  and  $xy=y$  respectively.

*Theorem 1.3.2.* The relation of inclusion is reflexive, anti-symmetric and transitive, i.e., it is a partial order in  $B$ .

*Proof.* Reflexive law:  $x \leq x$ . Since  $x+x=x$ , we have  $x \leq x$ .

Anti-symmetric law: If  $x \leq y$  and  $x \geq y$ , then  $x=y$ .  $x = x+y=y$ .

Transitive law: If  $x \leq y$  and  $y \leq z$ , then  $x \leq z$ .  $x+z = x+y+z = y+z = z$ . Hence  $z \leq x$ .

*Theorem 1.3.3.* 0 is the least element and 1 is the greatest element with respect to the partial order  $\leq$ .

*Proof.* Since  $x+0=x$ , we have  $0 \leq x$  and  $1 \geq x$  dually.

*Theorem 1.3.4.* The one-to-one mapping  $x \rightarrow x'$  from  $B$  onto  $B$  is a dual-automorphism of  $B$ .

*Proof.* What is to be proved is that  $(x')' = x$ , and  $x' \geq y'$  if and only if  $x \leq y$ . Now assume  $x \leq y$ , then we have  $x+y=y$  which, when complemented, yields  $x'y' = y'$ . It follows that  $x' \geq y'$ . The converse will be proved similarly.  $(x')' = x$  was already proved.

*Theorem 1.3.5.*  $B$  is a complemented distributive lattice.

*Proof.* Since  $x(x+y)=x$ , we have  $x \leq x+y$  and, by symmetry,  $y \leq x+y$ . Now assume  $x \leq z$  and  $y \leq z$ , then  $x+y+z = (x+z) + (y+z) = z+z = z$ . It follows that

$x + y \leq z$ . Hence  $x + y$  is the least upper bound or join of  $x$  and  $y$ , i.e.,  $x + y = x \vee y$ . Dually  $xy$  is the greatest lower bound or meet of  $x$  and  $y$ , i.e.,  $xy = x \wedge y$ . Thus it has been shown that  $B$  is a lattice. Its complementedness and distributivity are evident from Postulates 4, Postulate 5 and Theorem 1.2.5.

#### 1.4. Boolean Algebras as Rings

In the study of the theory as well as the applications of Boolean algebras, some operations other than addition, multiplication and complementation are frequently of use. As such auxiliary operations, we here consider the three operations called *Peirce's operation*, *Sheffer's operation* and *the ring addition*. Their notations and definitions are given as follows:

Peirce's operation:  $x \downarrow y = x'y'$ .

Sheffer's operation:  $x|y = x' + y'$ .

Ring addition:  $x \oplus y = x'y + xy' = (x + y)(x' + y')$ <sup>1)</sup>.

The first two operations are interesting in the sense that each of them alone can express the three fundamental operations. Indeed, we have

$$\begin{aligned} x' &= x'x' = x \downarrow x, \\ x + y &= (x'y')' = (x \downarrow y) \downarrow (x \downarrow y), \\ xy &= (x')'(y')' = (x \downarrow x) \downarrow (y \downarrow y) \end{aligned}$$

for Peirce's operation and

$$\begin{aligned} x' &= x' + x' = x|x, \\ x + y &= (x')' + (y')' = (x|x)|(y|y), \\ xy &= (x' + y')' = (x|y)|(x|y) \end{aligned}$$

for Sheffer's operation. As a consequence, every Boolean expression can be expressed by any of the two operations alone. Besides this theoretical significance, they are fundamental in the application of Boolean algebras to vacuum tube and transistor switching circuits.

The ring addition has no such property as the other two, but it is rather more important in other respects. The rest of the section will be devoted to the description of its properties and their consequences.

*Theorem 1.4 4.  $B$  is a commutative ring with respect to the ring addition and the multiplication.*

*Proof.* The ring addition is obviously commutative. Its associativity is proved as follows.

$$\begin{aligned} (x \oplus y) \oplus z &= (x \oplus y)'z + (x \oplus y)z' = (x'y + xy')'z + (x'y + xy')z' \\ &= (x'y' + xy)z + (x'y + xy')z' = (x'y'z + xyz) + (x'yz' + xy'z') \\ &= (x'y'z + x'yz') + (xy'z' + xyz) = x'(y'z + yz') + x(y'z' + yz) \\ &= x'(y'z + yz') + x(y'z + yz')' = x'(y \oplus z) + x(y \oplus z)' = x \oplus (y \oplus z). \end{aligned}$$

<sup>1)</sup> We adopt the symbol  $\oplus$  following the latest engineering trend. Some writers have used this symbol for other purposes, but there is no fear of confusion.

Hence we can write ring sums of any number of elements unambiguously without parentheses.

Now observe that  $x \oplus 0 = x$  and  $x \oplus x = 0$  for every  $x$ , because  $x \oplus 0 = x' \cdot 0 + x \cdot 0' = 0 + x \cdot 1 = 0 + x = x$  and  $x \oplus x = x'x + xx' = 0 + 0 = 0$ . Accordingly,  $B$  is an additive group (Abelian group), 0 being the zero element and any element the negative of itself. Furthermore, the distributive law holds between the ring addition and the multiplication, i.e.,  $x(y \oplus z) = xy \oplus xz$ , because  $x(y \oplus z) = x(y'z + yz') = xy'z + xy'z' = (0 + xy'z) + (0 + xy'z') = (x'yxz + xy'z) + (xx'y + xy'z') = (x' + y')xz + xy(x' + z') = (xy)'xz + xy(xz)' = xy \oplus xz$ .

Since the multiplication is commutative and associative, the theorem is proved.

Note here that  $B$  has the unit 1 ( $1 \cdot x = x$ ), and every element of it is idempotent ( $xx = x$ ), or equivalently,  $B$  is a *Boolean ring with unit*<sup>1)</sup>. Further note that the addition and the complementation can be expressed by means of the ring addition and the multiplication as  $x + y = x \oplus y \oplus xy$  and  $x' = 1 \oplus x$  respectively. The proof is as follows.

$$\begin{aligned} x \oplus y \oplus xy &= (x \oplus y)'xy + (x \oplus y)(xy)' = (x'y + xy')'xy + (x'y + xy')(xy)' \\ &= (x'y' + xy)xy + (x'y + xy')(x' + y') = xy + x'y + xy' = x(y + y') + x'y \\ &= x + x'y = x + y. \\ 1 \oplus x &= 1' \cdot x + 1 \cdot x' = 0 \cdot x + x' = 0 + x' = x'. \end{aligned}$$

*Theorem 1.4.2. Let  $B$  be a Boolean ring with the unit 1, then, if we define  $x + y = x \oplus y \oplus xy$  and  $x' = 1 \oplus x$ ,  $B$  becomes a Boolean algebra.*

*Proof.* Since any element of  $B$  is idempotent, we have:

$$x \oplus y = (x \oplus y)(x \oplus y) = xx \oplus xy \oplus yx \oplus yy = x \oplus xy \oplus yx \oplus y = (x \oplus y) \oplus (xy \oplus yx),$$

and hence,  $xy \oplus yx = 0$ . Now, put  $y = x$  in the last equation, then we obtain  $x \oplus x = 0$ , which, in turn, yields  $xy = yx$ , because  $xy = xy \oplus 0 = xy \oplus xy \oplus yx = 0 \oplus yx = yx$ . Thus, the commutative law of multiplication (Postulate 3 b) is proved. Postulates 1, 2, and 6 being evident from the beginning, the remaining task is to prove Postulates 3 a, 4 and 5.

*Postulate 3 a:*  $x + y = x \oplus y \oplus xy = y \oplus x \oplus yx = y + x$ .

*Postulate 4 a:*  $x(y + z) = x(y \oplus z \oplus yz) = xy \oplus xz \oplus xyz = xy \oplus xz \oplus (xy)(xz) = xy + xz$ .

*Postulate 4 b:*  $(x + y)(x + z) = (x \oplus y \oplus xy)(x \oplus z \oplus xz)$   
 $= xx \oplus xz \oplus xz \oplus xy \oplus yz \oplus yxz \oplus xyx \oplus xyz \oplus yxz$   
 $= x \oplus xz \oplus xz \oplus xy \oplus yz \oplus xyz \oplus xy \oplus xyz \oplus xyz$   
 $= x \oplus (xz \oplus xz) \oplus (xy \oplus xy) \oplus yz \oplus xyz \oplus (xyz \oplus xyz)$   
 $= x \oplus yz \oplus xyz = x + yz$ .

*Postulate 5:*  $xx' = x(1 \oplus x) = x \oplus xx = x \oplus x = 0$ .

$$x + x' = x \oplus x' \oplus xx' = x \oplus x' \oplus 0 = x \oplus x' = x \oplus 1 \oplus x = 1 \oplus x \oplus x = 1 \oplus 0 = 1.$$

<sup>1)</sup> cf. Ref. 39,

This completes the proof.

By the above two theorems, it has been shown that there is a one-to-one correspondence between Boolean algebras and Boolean rings with unit.

### 1.5. Examples of Boolean Algebras

So far we have described fundamental concepts of Boolean algebras from a purely formal point of view. In this section, we shall present several examples to which the concepts of Boolean algebras can be actually applied.

#### 1.5.1. Binary Functions

Let us consider the class  $F$  of all real-valued functions  $x(\omega), y(\omega), \dots$  which are defined on a space  $I$  of points  $\omega$  and can take the values 0 and 1, and define:

$$\begin{aligned} x+y: & \text{Max}(x, y), \\ xy: & \text{Min}(x, y), \\ 0: & \text{the function which is 0 all over } I, \\ 1: & \text{the function which is 1 all over } I, \\ x': & 1-x. \end{aligned}$$

With these definitions, the postulates of Boolean algebras are satisfied, and hence,  $F$  becomes a Boolean algebra. The proper interpretation of the relation of inclusion and the ring addition is given by:

$$x \leq y: x \text{ is equal to or smaller than } y,$$

and

$$x \oplus y: x+y \pmod{2}.$$

respectively, where the symbol  $+$  means ordinary addition. Owing to this interpretation, the ring addition is called the *addition modulo 2* sometimes.

#### 1.5.2. Set Calculus

Consider the class  $S$  of all subsets  $X, Y, \dots$  of an aggregate  $I$  and define:

$$\begin{aligned} X+Y \ (X \cup Y): & \text{the set of points } \omega \text{ such that } \omega \in X \text{ or } \omega \in Y, \\ X \cdot Y \ (X \cap Y): & \text{the set of points } \omega \text{ such that } \omega \in X \text{ and } \omega \in Y, \\ 0 \ (\emptyset): & \text{the empty set,} \\ 1 \ (I): & \text{the whole space,} \\ X' \ (X^c): & \text{the set of points such that } \omega \notin X. \end{aligned}$$

Thus interpreted,  $S$  becomes a Boolean algebra where the proper interpretation of  $X \leq Y$  and  $X \oplus Y$  is given by:

$$X \leq Y \ (X \subset Y): "Y \text{ includes } X",$$

and

$$X \oplus Y \ (X \triangle Y): (X^c \cap Y) \cup (X \cap Y^c)$$

respectively. The name "inclusion" for the relation  $\leq$  has its origin in this interpretation. The set  $X \triangle Y$  is called the *symmetric difference* of  $X$  and  $Y$  because it is the union of the two *difference sets*,  $Y-X = X^c \cap Y$  and  $X-Y = X \cap Y^c$ .



As is well known, there is a close relationship between  $S$  and  $F$ . That is: To each  $X$  of  $S$ , there corresponds a unique  $x(\omega)$  of  $F$  such that  $x(\omega)=1$  for every  $\omega \in X$  and  $x(\omega)=0$  for every  $\omega \notin X$ . In this correspondence, the function  $x(\omega)$  is called the *characteristic function* of the subset  $X$ .

The fact that the set calculus obeys the laws of Boolean algebras provides an easy and quick method for proving or remembering the theorems of Boolean algebras, because the corresponding theorems of set calculus can be proved or remembered intuitively by the so-called *Venn diagrams*. As an illustration, let us consider the theorems  $xy(x \oplus y) = 0$  and  $xy + (x \oplus y) = x + y$ . The corresponding theorems of set calculus are  $(X \cap Y) \cap (X \triangle Y) = \emptyset$  and  $(X \cap Y) \cup (X \triangle Y) = X \cup Y$ . The venn diagrams of the sets  $X \cap Y$ ,  $X \cup Y$  and  $X \triangle Y$  are given in Fig. 1.5.1. A glance at these diagrams is enough to conclude that the intersection of  $X \cap Y$  and  $X \triangle Y$  is empty and their union is equal to  $X \cup Y$ .

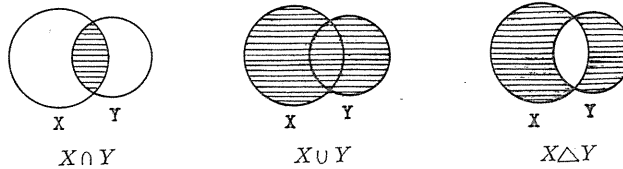


FIG. 1.5.1

### 1.5.3. Calculus of Propositions

Consider a class  $P$  of propositions  $x, y, \dots$ , where a proposition is a sentence which is either true or false. We now define *logical sum*, *logical product*, *falsehood*, *truth* and *negative* as follows:

- $x + y$ : " $x$  or  $y$ ", a proposition which is true if and only if at least either of  $x$  or  $y$  is true,
- $x \cdot y$ : " $x$  and  $y$ ", a proposition which is true if and only if both  $x$  and  $y$  are true,
- $0$ : "falsehood", a proposition which is always false,
- $1$ : "truth", a proposition which is always true,
- $x'$ : "not  $x$ ", a proposition which is true if and only if  $x$  is false.

With these definitions,  $P$  becomes a Boolean algebra. The proper interpretation of the relation  $x \leq y$  is " $x$  implies  $y$ ", or more exactly, " $x$  is false or  $x$  and  $y$  are true". Consequently, for example, the theorems "If  $x \leq y$  and  $y \leq z$ , then  $x \leq z$ " and " $x' \geq y'$  if and only if  $x \leq y$ " are interpreted as "If  $x$  implies  $y$  and  $y$  implies  $z$ , then  $x$  implies  $z$ " and "Not  $y$  implies not  $x$  if and only if  $x$  implies  $y$ " respectively. On the other hand  $x \oplus y$  is read as "*either  $x$  or else  $y$* " and is interpreted as a proposition which is true if and only if either  $x$  is false and  $y$  is true or  $x$  is true and  $y$  is false. On account of this meaning of  $x \oplus y$  in calculus of propositions, the operation  $\oplus$  is called the *exclusive or*.

Calculus of propositions is indeed the origin of Boolean algebras. George Boole, the founder, treated calculus of propositions in his first account for Boolean algebras published in 1854.

## 2. Theory of Boolean Functions

### 2.1. Boolean Functions

Consider  $n$  elements  $x_1, x_2, \dots, x_n$  of a Boolean algebra  $B$ , and assume that they are mutually independent, or equivalently, that there is no relation among them other than those derived from the postulates of  $B$ . Any expression formed from these elements through Boolean operations is called a *Boolean function of  $n$  variables*. The class of all Boolean functions of  $n$  variables forms a subalgebra  $B_n$  of  $B$ .  $B_n$  is called a *free Boolean algebra generated by  $n$  elements*<sup>1)</sup>. In this chapter, the investigations will be mainly concerned with the structure of  $B_n$  and Boolean functions.

The simplest Boolean functions are the variables  $x_i$  themselves and their complements  $x'_i$ . These functions are called collectively as *literals*. Literals are denoted conveniently by the symbols  $x_i^{\alpha_i}$  where  $\alpha_i$  belongs to a two-element Boolean algebra  $B_0$  and  $x_i^{\alpha_i} = x_i$  if  $\alpha_i = 1$  and  $x_i^{\alpha_i} = x'_i$  if  $\alpha_i = 0$ .

Functions of the form  $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n} (x_1^{\beta_1} + x_2^{\beta_2} + \cdots + x_n^{\beta_n})$  are called *atoms* (*anti-atoms*). There are  $2^n$  atoms (anti-atoms) and they are denoted by the symbols  $a_i = x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n} (b_i = x_1^{\beta_1} + x_2^{\beta_2} + \cdots + x_n^{\beta_n})$  where  $i$  is the integer whose binary expression is given by  $\alpha_1 \alpha_2 \cdots \alpha_n (\beta'_1 \beta'_2 \cdots \beta'_n)$ . Thus, for example, the four atoms (anti-atoms) of  $B_2$  are  $a_0 = x'_1 x'_2$ ,  $a_1 = x'_1 x_2$ ,  $a_2 = x_1 x'_2$  and  $a_3 = x_1 x_2$  ( $b_0 = x_1 + x_2$ ,  $b_1 = x_1 + x'_2$ ,  $b_2 = x'_1 + x_2$  and  $b_3 = x'_1 + x'_2$ ). As is easily seen from De Morgan's laws,  $a_i$  and  $b_i$  are the complement of one another for every  $i$ .

*Lemma 2.1.1. For any two atoms  $a_i$  and  $a_j$ , either  $a_i a_j = a_i = a_j$  or  $a_i a_j = 0$ .*

*Proof.* If  $a_i = a_j$ , then  $a_i a_j = a_i a_i = a_i = a_j$ . If, on the contrary,  $a_i \neq a_j$ , then, putting  $a_i = x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$  and  $a_j = x_1^{\beta_1} x_2^{\beta_2} \cdots x_n^{\beta_n}$ , we have  $a_i a_j \leq x_k^{\alpha_k} x_k^{\beta_k}$  for all  $k$ . But  $x_k^{\alpha_k} x_k^{\beta_k} = 0$ , being  $x_k x'_k$  or  $x'_k x_k$ , for at least one  $k$ . Hence  $a_i a_j = 0$ .

We shall now prove a fundamental theorem characterizing the structure of  $B_n$ .

*Theorem 2.1.1. Every Boolean function  $f(x_1, x_2, \dots, x_n)$  can be uniquely represented by a sum of atoms as:*

$$f(x_1, x_2, \dots, x_n) = \sum f(\alpha_1, \alpha_2, \dots, \alpha_n) x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n},$$

*and, dually, every Boolean function  $f(x_1, x_2, \dots, x_n)$  can be uniquely represented by a product of anti-atoms as:*

$$f(x_1, x_2, \dots, x_n) = \prod [f(\alpha_1, \alpha_2, \dots, \alpha_n) + x_1^{\alpha'_1} + x_2^{\alpha'_2} + \cdots + x_n^{\alpha'_n}].$$

*Proof.* Let  $I$  be the set of integers  $[0, 1, \dots, 2^n - 1]$ . Associate with each non-empty subset  $S$  of  $I$ , the function  $g_S = \sum_{i \in S} a_i$  and define  $g_\emptyset = 0$ .

We shall prove: (1)  $g_S = g_T$  if and only if  $S = T$ , (2)  $0 = g_\emptyset$ ,  $1 = g_I$ ,  $g_S + g_T = g_{S \cup T}$ ,  $g_S g_T = g_{S \cap T}$  and  $g'_S = g_{S^c}$ , (3) every Boolean function is represented by a  $g_S$ .

*Proof of (1):* The "if" part being evident, it suffices to prove the "only if"

<sup>1)</sup> cf. Ref. 1.

part. To begin with, it is noted by Lemma 2.1.1 that  $a_i g_s = a_i$  if and only if  $i \in S$  and  $a_i g_s = 0$  if and only if  $i \notin S$ . Now assume  $g_s = g_t$ , then, for each  $i \in S$ ,  $a_i = a_i g_s = a_i g_t$ , whence  $i \in T$ . It follows that  $S \subset T$  and, by symmetry,  $S \subset T$ . Therefore  $S = T$ .

*Proof of (2):* By definition  $0 = g_\emptyset$ , while, by the distributive law,  $1 = \prod_{j=1}^n (x_j + x'_j) = \sum_{i \in I} a_i = g_I$ . Next,  $g_s + g_t = \sum_{i \in s} a_i + \sum_{i \in t} a_i = \sum_{i \in s \cup t} a_i + \sum_{i \in s \cap t} a_i = g_{s \cup t} + g_{s \cap t}$ . But clearly  $g_{s \cup t} \geq g_{s \cap t}$ , i.e.,  $g_{s \cup t} + g_{s \cap t} = g_{s \cup t}$ , whence  $g_s + g_t = g_{s \cup t}$ . Again, by the distributive law,  $g_s g_t = \sum_{i \in s} a_i \cdot \sum_{j \in t} a_j = \sum_{i \in s} \sum_{j \in t} a_i a_j$ . Therefore, by Lemma 2.1.1,  $g_s g_t = \sum_{i \in s \cap t} a_i = g_{s \cap t}$ . It follows that  $g_s + g_{s^c} = g_{s \cup s^c} = g_I = 1$  and  $g_s g_{s^c} = g_{s \cap s^c} = g_\emptyset = 0$ , whence  $g'_s = g_{s^c}$ .

*Proof of (3):* By (2), every Boolean function of  $g_s$  is in itself a  $g_s$ . Therefore it is sufficient to show that every  $x_j$  is a  $g_s$ . But, if  $X_j$  denotes the set of  $i = \alpha_1 \alpha_2 \cdots \alpha_n$  such that  $\alpha_j = 1$ , then  $x_j = x_j \prod_{k \neq j} (x_k + x'_k) = \sum_{i \in X_j} a_i = g_{X_j}$ .

Let us now assume that a function  $f$  is represented by  $a$  sum of atoms, and write:

$$f(x_1, x_2, \dots, x_n) = \sum_i f_i a_i = \sum_i f_{a_1 a_2 \dots a_n} x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n},$$

where  $f_i = f_{a_1 a_2 \dots a_n}$  is 1 if  $a_i$  is a summand of  $f$  and is 0 otherwise. Since the above relation is an identity, any function can be substituted for any variable. If we put  $x_1 = \alpha_1, x_2 = \alpha_2, \dots, x_n = \alpha_n$ , then the atom  $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$  becomes 1 while all other atoms become 0. It follows that  $f(\alpha_1, \alpha_2, \dots, \alpha_n) = f_{a_1 a_2 \dots a_n}$ . Thus the first half of the theorem is proved.

The proof of the second half is as follows. Associate with each non-empty subset  $S$ , the function  $h_s = \prod_{i \in s} b_i$  and define  $h_\emptyset = 1$ . Then, by (2) and  $a_i = b_i$ , we obtain  $g_s = h_{s^c}$ . Hence it follows from (3) that every Boolean function is represented by a  $h_s$ . Furthermore, this representation is unique by (1). Now, for any function  $f$ , we have:

$$f'(x_1, x_2, \dots, x_n) = \sum_i f'(\alpha_1, \alpha_2, \dots, \alpha_n) x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n},$$

which, when complemented, yields the representation:

$$f(x_1, x_2, \dots, x_n) = \prod [f(\alpha_1, \alpha_2, \dots, \alpha_n) + x_1^{\alpha_1'} + x_2^{\alpha_2'} + \cdots + x_n^{\alpha_n'}].$$

This completes the proof.

We can see from the proof of the theorem that  $B_n$  is isomorphic with the algebra of all subsets of  $2^n$  points and hence the total number of Boolean functions of  $n$  variables is  $2^{2^n}$ . The sum (product) of atoms (anti-atoms) representing a function  $f$  is called the *standard sum (product)* of  $f$ . When  $f = g_s = h_{s^c}$ , where  $S = [a, b, \dots, k]$  and  $S^c = [a', b', \dots, k']$ , the standard sum and the standard product of  $f$  can be conveniently denoted by:

$$f = \sum (a, b, \dots, k),$$

and

$$f = \prod (a', b', \dots, k')$$

respectively.

*Example 2.1.1.* Represent the function:  $f(x, y, z) = xy + xz + yz$  by the standard sum and the standard product.

*Solution.* The eight values  $f(\alpha_1, \alpha_2, \alpha_3)$  are easily evaluated and are given in the following truth table. Thus, the standard sum is given by:

$$f = x'yz + xy'z + xyz' + xyz = \sum (3, 5, 6, 7),$$

and the standard product by:

$$\begin{aligned} f &= (x' + y + z)(x + y' + z)(x + y + z')(x + y + z) \\ &= \prod (0, 1, 2, 4). \end{aligned}$$

In connection with Theorem 2.1.1, an important concept, *dimension*, is introduced. The number of atoms appearing in the standard sum of a function  $f$  is called the dimension of  $f$  and is denoted by the symbol  $d(f)$ . As is easily seen, the dimension has the properties: (1)  $0 \leq d(f) \leq 2^n$ , (2)  $d(f) = 0$  if and only if  $f = 0$ , and  $d(f) = 2^n$  if and only if  $f = 1$ , (3)  $d(f) \leq d(g)$  if  $f \leq g$  and (4)  $d(f) + d(g) = d(f + d) + d(fg)$ , in particular,  $d(f) + d(f') = 2^n$ .

Atoms are characterized by the property that  $d(f) = 1$  if and only if  $f$  is an atom. By the way, we shall show another characterization of atoms for later uses.

*Lemma 2.1.2.* A Boolean function  $a$  is an atom if and only if  $a \neq 0$  and either  $fa = 0$  or  $fa = a$  for every Boolean function  $f$ .

*Proof.* The "only if" part being evident, we shall prove the "if" part only. Let us assume that a function  $g$  is neither 0 nor an atom. Then  $g$  is a sum of at least two atoms. Let  $a$  be an atom which is a summand of  $g$ . Now, putting  $f = a$ , we obtain  $fg = a$ . But obviously  $a \neq 0$  and  $a \neq g$ . This completes the proof.

*Theorem 2.1.2.* For any set of variables, say,  $[x_1, x_2, \dots, x_n]$ , every Boolean function  $f$  can be uniquely represented by a sum of the form:

$$f(x_1, x_2, \dots, x_n) = \sum x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_s^{\alpha_s} f(\alpha_1, \alpha_2, \dots, \alpha_s, x_{s+1}, \dots, x_n),$$

and dually by a product of the form:

$$f(x_1, x_2, \dots, x_n) = \prod [x_1^{\alpha_1} + x_2^{\alpha_2} + \cdots + x_s^{\alpha_s} + f(\alpha'_1, \alpha'_2, \dots, \alpha'_s, x_{s+1}, \dots, x_n)].$$

*Proof.* By duality, it suffices to prove the first half. Since, by Theorem 2.1.1,

$$\begin{aligned} f(x_1, x_2, \dots, x_n) &= \sum x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n} f(\alpha_1, \alpha_2, \dots, \alpha_n) \\ &= \sum x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_s^{\alpha_s} \sum x_{s+1}^{\alpha_{s+1}} \cdots x_n^{\alpha_n} f(\alpha_1, \alpha_2, \dots, \alpha_n), \end{aligned}$$

TABLE 2.1.1

	$x$	$y$	$z$	$f(x, y, z)$
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

the possibility and the uniqueness of the representation are obvious. Now assume that  $f$  is represented by:

$$f(x_1, x_2, \dots, x_n) = \sum x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_s^{\alpha_s} f_{\alpha_1 \alpha_2 \dots \alpha_s}(x_{s+1}, \dots, x_n).$$

Then, substituting  $x_1 = \alpha_1, x_2 = \alpha_2, \dots, x_s = \alpha_s$ , we obtain:

$$f(\alpha_1, \alpha_2, \dots, \alpha_s, x_{s+1}, \dots, x_n) = f_{\alpha_1 \alpha_2 \dots \alpha_s}(x_{s+1}, \dots, x_n).$$

This completes the proof.

**Theorem 2.1.3.** For any set of variables, say,  $[x_1, x_2, \dots, x_s]$ , and any Boolean function  $f$ ,

$$x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_s^{\alpha_s} f(x_1, x_2, \dots, x_n) = x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_s^{\alpha_s} f(\alpha_1, \alpha_2, \dots, \alpha_s, x_{s+1}, \dots, x_n),$$

and dually

$$\begin{aligned} x_1^{\alpha_1} + x_2^{\alpha_2} + \cdots + x_s^{\alpha_s} + f(x_1, x_2, \dots, x_n) \\ = x_1^{\alpha_1} + x_2^{\alpha_2} + \cdots + x_s^{\alpha_s} + f(\alpha_1', \alpha_2', \dots, \alpha_s', x_{s+1}, \dots, x_n). \end{aligned}$$

*Proof.* From Theorem 2.1.2, we have:

$x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_s^{\alpha_s} f(x_1, x_2, \dots, x_n) = x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_s^{\alpha_s} \sum x_1^{\beta_1} x_2^{\beta_2} \cdots x_s^{\beta_s} f(\beta_1, \beta_2, \dots, \beta_s, x_{s+1}, \dots, x_n)$ . But  $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_s^{\alpha_s} x_1^{\beta_1} x_2^{\beta_2} \cdots x_s^{\beta_s}$  is  $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_s^{\alpha_s}$  if  $\alpha_1 = \beta_1, \alpha_2 = \beta_2, \dots, \alpha_s = \beta_s$ , and is 0 otherwise. Therefore the first half is proved. The second half is obvious from the duality.

In Section 1.1.4. we have seen that a Boolean algebra in a Boolean ring with unit. Then, what can be said about the structure of the special Boolean algebra  $B_n$  as a ring?

**Theorem 2.1.4.** Let  $S(f)$  be the set of all Boolean functions  $g$  such that  $g \leq f$  for a function  $f$ , then  $S(f)$  is an ideal of  $B_n$ . Conversely, every ideal of  $B_n$  is a principal ideal of the form  $S(f)$ .

*Proof.* Let  $g$  and  $h$  be any two elements of  $S(f)$ , then

$$g \oplus h = gh' + g'h \leq g + h \leq f + f = f,$$

i.e.,  $g \oplus h$  is an element of  $S(f)$ . Again, let  $g$  be an element of  $S(f)$  and  $k$  be any element of  $B_n$ , then

$$gk \leq g \leq f,$$

i.e.,  $gk$  is an element of  $S(f)$ . It follows that  $S(f)$  is an ideal of  $B_n$ .

Now assume that  $S$  is an ideal of  $B_n$ . Then, since  $S$  is a sublattice of the finite lattice  $B_n$ , it has the greatest element, say,  $f$ . Hence  $S \subset S(f)$ . While, for any element  $g$  of  $S(f)$ ,  $g = gf \in S$ . Therefore  $S \subset S(f)$ . It follows that  $S = S(f)$ .

The theorem indicates that there is a one-to-one correspondence:  $f \leftrightarrow S(f)$  between  $B_n$  and the class of all ideals of  $B_n$ . The ideals corresponding to atoms are called *atomic ideals*. Clearly atomic ideals and only these are simple ideals.

*Theorem 2.1.5.*  $B_n$  is the direct sum of all atomic ideals.

*Proof.* What is to be proved is that every Boolean function can be uniquely represented by a ring sum of atoms. But this is evident from Theorem 2.1.1 and the fact that  $f_1 + f_2 + \cdots + f_m = f_1 \oplus f_2 \oplus \cdots \oplus f_m$  if  $f_i f_j = 0$  for every  $i \neq j$ .

## 2.2. Linear Functions

Boolean functions which can be formed from  $n$  variables  $x_1, x_2, \dots, x_n$  and their complements through the ring additions only are called *linear functions*. In other words, a linear function is a Boolean function of the form:

$$\beta_1 x_1^{\alpha_1} \oplus \beta_2 x_2^{\alpha_2} \oplus \cdots \oplus \beta_n x_n^{\alpha_n},$$

where  $\alpha$ 's and  $\beta$ 's are elements of a two-element Boolean algebra  $B_0$ . Making use of the formula  $x' = 1 \oplus x$ , every linear function is uniquely represented in the *standard form*:

$$\beta_0 \oplus \beta_1 x_1 \oplus \cdots \oplus \beta_n x_n.$$

For any linear function, the number of  $i > 0$  such that  $\beta_i = 1$  is called its *length*. Thus, for example, the length of 0 and 1 is 0, and that of  $x'_1 \oplus x_2 \oplus x_3$  is 3.

*Theorem 2.1.2.* Every linear function other than 0 and 1 has the dimension  $2^{n-1}$ .

*Proof.* First consider a linear function  $f$  with the standard form  $x_1 \oplus x_2 \oplus \cdots \oplus x_k$ , and prove that  $f = \sum' x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_k^{\alpha_k}$ , where  $\sum'$  means the summation taken for all products  $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_k^{\alpha_k}$  such that the numbers of  $x_i$  with  $\alpha_i = 1$  are odd. The proof can be carried out by a mathematical induction on the length  $k$  as follows: The proposition to be proved is obviously valid for  $k = 1$ . Now assume that it is valid for  $k = m$ . Then we have:  $x_1 \oplus x_2 \oplus \cdots \oplus x_m \oplus x_{m+1} = (x_1 \oplus x_2 \oplus \cdots \oplus x_m)' x_{m+1} + (x_1 \oplus x_2 \oplus \cdots \oplus x_m) x_{m+1}$

$$= x_{m+1} \sum'' x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_m^{\alpha_m} + x_{m+1}' \sum' x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_m^{\alpha_m} = \sum' x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_m^{\alpha_m} x_{m+1}^{\alpha_{m+1}},$$

where  $\sum''$  means the summation taken for all products  $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_m^{\alpha_m}$  such that the numbers of  $x_i$  with  $\alpha_i = 1$  are even. Hence it is also valid for  $k = m + 1$ . Thus, it has been proved that  $f$  is a sum of  $2^{k-1}$  products  $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_k^{\alpha_k}$ . But each of these products is a sum of  $2^{n-k}$  atoms  $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_k^{\alpha_k} x_{k+1}^{\alpha_{k+1}} \cdots x_n^{\alpha_n}$ . It follows then that the dimension of  $f$  is  $2^{k-1} 2^{n-k} = 2^{n-1}$ .

Next, consider a linear function with the standard form  $1 \oplus x_1 \oplus x_2 \oplus \cdots \oplus x_k$ . Then, since it is the complement of the above  $f$ , its dimension is given by  $2^n - 2^{n-1} = 2^{n-1}$ . This completes the proof.

In view of the fact found in the above proof, we call linear functions with  $\beta_0 = 0$  as *odd linear functions* and those with  $\beta_0 = 1$  as *even linear functions*. Evidently the complement of an odd linear function is an even linear function and *vice versa*.

The totality of linear functions of  $n$  variables forms a commutative group

$L_n^*$  with respect to the operation  $\oplus$ , and its subset consisting of all odd linear functions forms its subgroup  $L_n$ . Clearly the order of  $L_n^*$  is  $2^{n+1}$  and that of  $L_n$  is  $2^n$ .

A number of distinct odd linear functions  $y_1, y_2, \dots, y_m$  are said to be *mutually independent* if and only if  $\beta_1 y_1 \oplus \beta_2 y_2 \oplus \dots \oplus \beta_m y_m = 0$  implies  $\beta_1 = \beta_2 = \dots = \beta_m = 0$ , where  $\beta_i$ 's belong to  $B_0$ . A number of distinct linear functions  $y_1^{\alpha_1}, y_2^{\alpha_2}, \dots, y_m^{\alpha_m}$  are said to be *mutually independent* if and only if the corresponding odd linear functions  $y_1, y_2, \dots, y_m$  are mutually independent. As will be easily seen, there is no set of more than  $n$  mutually independent (odd) linear functions, but there are sets of  $n$  or less than  $n$  mutually independent (odd) linear functions. For example,  $[x_1, x_2, \dots, x_n]$  is such a set.

**Theorem 2.2.2.** *There are  $\prod_{i=0}^{k-1} (2^n - 2^i)/k!$  distinct sets of  $k$  mutually independent odd linear functions and  $2^k \prod_{i=0}^{k-1} (2^n - 2^i)/k!$  distinct sets of  $k$  mutually independent linear functions for every  $k \leq n$ .*

*Proof.* It is sufficient to show that there are  $\prod_{i=0}^{k-1} (2^n - 2^i)$  distinct ordered  $k$ -tuples of mutually independent odd linear functions, because, to each set of  $k$  distinct odd linear functions  $[y_1, y_2, \dots, y_k]$ , there correspond  $k!$  distinct ordered  $k$ -tuples  $(y_a, y_b, \dots, y_s)$ , where  $(a, b, \dots, s)$  is a permutation of  $(1, 2, \dots, k)$ , and  $2^k$  distinct sets  $[y_1, y_2, \dots, y_k]$  of linear functions.

The proof can be carried out by a mathematical induction on the value of  $k$  as follows: First, it is obvious that there are  $(2^n - 1)(2^n - 2)$  ordered couples of mutually independent odd linear functions, because any two distinct odd linear functions other than 0 are mutually independent. Next, assume that the theorem is true for a certain  $k$ . Now observe: For each  $k$ -tuple  $(y_1, y_2, \dots, y_k)$ , the set of all odd linear functions of the form  $\beta_1 y_1 \oplus \beta_2 y_2 \oplus \dots \oplus \beta_k y_k$  constitutes a group  $L$  of the order  $2^k$  and  $y_1, y_2, \dots, y_k, y_{k+1}$  are mutually independent if and only if  $y_{k+1} \notin L$ . Accordingly,  $2^n - 2^k$   $k+1$ -tuples  $(y_1, y_2, \dots, y_k, y_{k+1})$  can be constructed from each  $k$ -tuple  $(y_1, y_2, \dots, y_k)$ . It follows that the theorem is also true for  $k+1$ . This completes the proof.

A mapping  $f \rightarrow \mu f$  from  $L_n (L_n^*)$  onto  $L_n (L_n^*)$  is said to be an *automorphism of  $L_n (L_n^*)$*  if and only if it is one-to-one and preserves the operation  $\oplus$ , or equivalently,  $\mu(f \oplus g) = \mu f \oplus \mu g$ . Now, let  $[y_1, y_2, \dots, y_n]$  be any set of  $n$  mutually independent odd linear functions, then it is evident that the mapping  $\beta_1 x_1 \oplus \beta_2 x_2 \oplus \dots \oplus \beta_n x_n \rightarrow \beta_1 y_1 \oplus \beta_2 y_2 \oplus \dots \oplus \beta_n y_n$  is an automorphism of  $L_n$  and the mapping  $\beta_0 \oplus \beta_1 x_1 \oplus \dots \oplus \beta_n x_n \rightarrow \beta_0 \oplus \beta_1 y_1^{\alpha_1} \oplus \dots \oplus \beta_n y_n^{\alpha_n}$  is an automorphism of  $L_n^*$ . Conversely, let  $\mu$  be an automorphism of  $L_n$ , then  $n$  odd linear functions  $\mu x_1, \mu x_2, \dots, \mu x_n$  are mutually independent since the independence is preserved under automorphisms. Thus, putting  $y_i = \mu x_i$  ( $i = 1, 2, \dots, n$ ),  $\mu$  can be written in the form  $\beta_1 x_1 \oplus \beta_2 x_2 \oplus \dots \oplus \beta_n x_n \rightarrow \beta_1 y_1 \oplus \beta_2 y_2 \oplus \dots \oplus \beta_n y_n$ . Similarly, any automorphism  $\mu$  of  $L_n^*$  must be of the form  $\beta_0 \oplus \beta_1 x_1 \oplus \dots \oplus \beta_n x_n \rightarrow \beta_0 \oplus \beta_1 y_1^{\alpha_1} \oplus \dots \oplus \beta_n y_n^{\alpha_n}$  where  $\mu x_i = y_i^{\alpha_i}$  ( $i = 1, 2, \dots, n$ ).

Now, as an immediate consequence of Theorem 2.2.2, we have:

**Theorem 2.2.3.** *There are  $\prod_{i=0}^{n-1} (2^n - 2^i)$  distinct automorphisms of  $L_n$  and  $2^n \prod_{i=0}^{n-1} (2^n - 2^i)$  distinct automorphisms of  $L_n^*$ .*

### 2.3. Automorphisms and Symmetries of $B_n$

A one-to-one mapping  $\mu: x \rightarrow \mu x$  from a Boolean algebra  $B$  onto  $B$  is called an *automorphism* of  $B$ , if and only if it preserves Boolean operations, i.e.,  $\mu(x+y) = \mu x + \mu y$ ,  $\mu(xy) = \mu x \cdot \mu y$  and  $\mu(x') = (\mu x)'$ . From the definition, it is evident that (1)  $\mu(x \oplus y) = \mu x \oplus \mu y$ , (2)  $\mu x \leq \mu y$  if and only if  $x \leq y$ , (3)  $\mu x = 0$  if and only if  $x = 0$ , (4)  $\mu x = 1$  if and only if  $x = 1$ .

Now we shall present a theorem characterizing automorphisms of  $B_n$ , the proof of which calls for a lemma.

**Lemma 2.3.1.** *For every automorphism  $\mu$  of  $B_n$ ,  $\mu a$  is an atom if and only if  $a$  is an atom.*

*Proof.* “If” part: Let  $a$  be an atom, then, from Lemma 2.1.2,  $a \neq 0$  and either  $fa = 0$  or  $fa = a$  for every Boolean function  $f$ . It follows that  $\mu a \neq 0$  and either  $g \cdot \mu a = 0$  or  $g \cdot \mu a = \mu a$  for every  $g$ , where we put  $g = \mu f$ . Hence, by the “if” part of the same lemma,  $\mu a$  must be an atom.

“Only if” part: Assume that  $\mu a$  is an atom, then  $\mu^{-1}(\mu a) = a$  where  $\mu^{-1}$  is the inverse of  $\mu$ . Therefore, by the “if” part,  $a$  must be an atom.

The lemma shows that atoms are permuted among themselves under an automorphism of  $B_n$ . Accordingly, the dimension is preserved under an automorphism of  $B_n$ ; for, if  $f = \sum a_i$  is the standard sum of  $f$ ,  $\mu f = \sum \mu a_i$  is the standard sum of  $\mu f$ .

**Theorem 2.3.1.** *Let  $a_i \rightarrow \mu a_i$  be a permutation of atoms, then the mapping  $\mu$  defined by  $\mu(\sum \beta_i a_i) = \sum \beta_i \mu a_i$  is an automorphism of  $B_n$ . Conversely, every automorphism of  $B_n$  is analytically expressed in a permutation of atoms induced by it.*

*Proof.* First half: Let  $f = \sum \beta_i a_i$  and  $g = \sum \gamma_i a_i$ , then  $\mu f = \sum \beta_i \mu a_i$  and  $\mu g = \sum \gamma_i \mu a_i$ . Since  $\mu a_i$  are atoms,  $\sum \beta_i \mu a_i$  and  $\sum \gamma_i \mu a_i$  represent the same function if and only if  $\beta_i = \gamma_i$  for all  $i$ , i.e.,  $f = g$ . Hence  $\mu$  is a one-to-one mapping. Further,  $\mu$  preserves Boolean operations, because

$$\begin{aligned} \mu(f+g) &= \mu(\sum \beta_i a_i + \sum \gamma_i a_i) = \mu(\sum (\beta_i + \gamma_i) a_i) = \sum (\beta_i + \gamma_i) \mu a_i \\ &= \sum \beta_i \mu a_i + \sum \gamma_i \mu a_i = \mu f + \mu g, \\ \mu(fg) &= \mu(\sum \beta_i a_i \cdot \sum \gamma_i a_i) = \mu(\sum \beta_i \gamma_i a_i) = \sum \beta_i \gamma_i \mu a_i = \sum \beta_i \mu a_i \cdot \sum \gamma_i \mu a_i \\ &= \mu f \cdot \mu g, \end{aligned}$$

and  $\mu(f') = \mu(\sum \beta_i' a_i) = \sum \beta_i' \mu a_i = (\sum \beta_i \mu a_i)' = (\mu f)'$ . Hence  $\mu$  is an automorphism of  $B_n$ .

Second half: Let  $\mu$  be an automorphism of  $B_n$ , then, by Lemma 2.3.1,  $a_i \rightarrow \mu a_i$  is a permutation of atoms. Since  $\mu$  is an automorphism, the image of  $f = \sum \beta_i a_i$



must be  $\mu f = \sum \beta_i \mu a_i$ . This completes the proof.

The theorem indicates that there is a one-to-one correspondence between the permutations of atoms and automorphisms of  $B_n$ . Therefore we have:

*Theorem 2.3.2. There are  $2^n!$  automorphisms of  $B_n$ .*

We now consider an important class of mappings from  $B_n$  onto  $B_n$  caused by permutations and/or complementations of the variables  $x_1, x_2, \dots, x_n$ . Obviously any of these mappings is an automorphism of  $B_n$ , since it induces a permutation of atoms. This class of automorphisms is called the *symmetry* of  $B_n$ .

The totality of symmetries forms a finite group  $O_n$ , the law of combination being defined by successive applications of symmetries. We adopt the usual cycle notation for permutations so that, for example  $\pi = (123)(45)$  means "replace  $x_1$  by  $x_2$ , replace  $x_2$  by  $x_3$ , replace  $x_3$  by  $x_1$  and interchange  $x_4$  with  $x_5$ ". The complementations of variables are denoted by operators  $\tau = [ab \dots k]$  which means "prime  $x_a, x_b, \dots, x_k$ ." Operators of the same kind as  $\tau$  are called *reflexion operators*. Here it is noted that, for any reflexion operator  $\tau = [ab \dots k]$ , and any permutation operator  $\pi$ ,  $\pi\tau\pi^{-1}$  is a reflexion operator and is given by  $\pi\tau\pi^{-1} = [\pi a, \pi b \dots \pi k]$ . When this property is used, every element  $\sigma$  of  $O_n$  can be written uniquely either in the form  $\sigma = \tau\pi$  or in the form  $\sigma = \pi\tau$ . Since there are  $2^n$  reflexion operators and  $n!$  permutation operators, the order of  $O_n$  is  $2^n n!$ . The group  $O_n$  is recognized as isomorphic to the group of symmetry operations of  $n$ -dimensional hypercube or hyperoctahedron and is called the *hyperoctahedral group*.

We now introduce the concepts of *type* and *genus* which are of primary importance for the classification of Boolean functions. Their definitions are as follows: *Two functions  $f$  and  $g$  are said to be congruent or of the same type if and only if there is a symmetry  $\sigma$  such that  $\sigma f = g$ . Two functions  $f$  and  $g$  are said to be of the same genus if and only if there is a symmetry  $\sigma$  such that either  $\sigma f = g$  or  $\sigma f' = g$ .* Thus, the concept of genus is wider than that of type. Ordinarily, a genus consists of two complementary types. But there are genera consisting of only one type, and it is clear that every function belonging to such a genus (type) is congruent with its own complement and consequently of the dimension  $2^{n-1}$ . Such functions (types or genera) are said to be *self-complementary*. For  $n \leq 3$ , every function of the dimension  $2^{n-1}$  is self-complementary, but, for  $n \geq 4$ , some functions of the dimension  $2^{n-1}$  are self-complementary and others are not, as will be shown in a later place<sup>1)</sup>.

By the well-known theory of finite group, the number of functions which are congruent with a given function  $f$  is equal to that of left cosets of the group  $G(f)$  where  $G(f)$  is the group formed by all symmetries  $\sigma$  such that  $\sigma f = f$  and is called the *symmetry group* of  $f$ . Hence it can be obtained by dividing the order of  $O_n$ ,  $2^n n!$ , by the order of  $G(f)$ .

A function whose symmetry group contains all permutation operators is said to be *symmetric* and a function which is congruent with a symmetric function to be *symmetrizable*. Thus, for example,  $x_1 x_2 x_3 + x'_1 x'_2 x'_3$  is symmetric and  $x_1 x_2 x'_3 + x'_1 x'_2 x_3$  is symmetrizable. The latter is sometimes said to be symmetric with respect to  $x_1, x_2$  and  $x'_3$ . The symmetric function which is the sum of all atoms  $x_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n}$  such that the number of variables  $x_j$  with  $\alpha_j = 1$  is

<sup>1)</sup> See page 93.

equal to  $i$  is called the *fundamental symmetric function* with the  $a$ -number  $i$ , and is denoted by  $S_i(x_1, x_2, \dots, x_n)$ . Clearly there are  $n+1$  fundamental symmetric functions  $S_i(x_1, x_2, \dots, x_n)$  ( $i=0, 1, \dots, n$ ) and their dimensions are  $\binom{n}{i}$  respectively. With these conventions, the following theorem is obvious from Theorem 2.1.1.

*Theorem 2.3.3. Every symmetric function  $f$  can be uniquely represented by a sum of fundamental symmetric functions:*

$$f(x_1, \dots, x_n) = S_a(x_1, \dots, x_n) + S_b(x_1, \dots, x_n) + \dots + S_k(x_1, \dots, x_n).$$

The right hand side of the above equation is called the *standard form* of symmetric function and is abbreviated as  $S_{a,b,\dots,k}(x_1, \dots, x_n)$ , where we agree that the  $a$ -numbers are arranged in the increasing order. Thus, for example,  $x_1 x_2 x_3 x_4 + x'_1 x'_2 x'_3 x'_4 = S_{0,4}(x_1, x_2, x_3, x_4)$ . With this notation for the standard form of symmetric function, we have the following theorems whose proofs may be omitted on account of their simplicity.

*Theorem 2.3.4.  $S'_{a,b,\dots,k}(x_1, \dots, x_n) = S_{a',b',\dots,k'}(x_1, \dots, x_n)$ , where  $[a, b, \dots, k]$  and  $[a', b', \dots, k']$  are the set-complement of one another with respect to the set  $[0, 1, \dots, n]$ .*

*Theorem 2.3.5.  $S_{a_1,\dots,a_k}(x_1, \dots, x_n) = \sum_{i=1}^k \sum_{b,c} S_b(x_1^*, \dots, x_r^*) S_c(x_{r+1}^*, \dots, x_n^*)$ , where  $[x_1^*, \dots, x_n^*]$  is a permutation of  $[x_1, \dots, x_n]$  and the first summation should be taken for all pairs of  $a$ -numbers  $b$  and  $c$  such that  $b+c=a_i$ ,  $0 \leq b \leq r$  and  $0 \leq c \leq n-r$ .*

Thus, for example, we have:

$$S'_{1,3,5}(x_1, x_2, x_3, x_4, x_5) = S_{0,2,4}(x_1, x_2, x_3, x_4, x_5)$$

$$\text{and} \quad S_{0,2,4}(x_1, x_2, x_3, x_4) = x'_1 S_{0,2}(x_2, x_3, x_4) + x_1 S_{1,3}(x_2, x_3, x_4).$$

When a function  $f$  is invariant under the reflexion operator priming the only one variable  $x_i$ ,  $f$  is said to be *independent* of the variable  $x_i$  and  $x_i$  to be *inessential* for  $f$ . A function is said to be *degenerate* or *non-degenerate* according to whether it has inessential variables or not. Of course, variables which are not inessential are said to be *essential*. Thus, a degenerate function is substantially a function of essential variables only and can be written in a form where all the inessential variables are absent. For example, the function  $x'_1 x_2 x'_3 + x'_1 x_2 x_3 + x_1 x'_2 x'_3 + x_1 x'_2 x_3$  is independent of the variable  $x_3$  and can be written as  $x_1 \oplus x_2$  where  $x_3$  is absent.

A function which is invariant under the reflexion operator priming all the variables is said to be *anti-self-dual*, since, by De Morgan's laws, its dual is equal to its complement. On the contrary, of course, a *self-dual* function is one which is turned into its complement by complementing all the variables. Evidently, a self-dual function is self-complementary.

A function whose symmetry group consists solely of the identity is said to be *perfectly asymmetric*. Perfectly asymmetric functions are rather rare for mode-

rate values of  $n$ . Indeed, as will be seen later<sup>1)</sup>, there are no perfectly asymmetric functions of three or less than three variables. But this is only a transitive phenomenon, and it has been found by Shannon<sup>2)</sup> that the proportion of perfectly asymmetric functions to all the functions of  $n$  variables tends to unity when  $n$  grows indefinitely.

#### 2.4. Functionally Separable Boolean Functions

A function  $f(x_1, \dots, x_n)$  is defined to be *functionally separable* if and only if it is non-degenerate and there are a function  $f^*$  of  $n-r+1$  variables and a function  $g$  of  $r$  variables such that

$$f(x_1, \dots, x_n) = f^*(g(x_1^*, \dots, x_r^*), x_{r+1}^*, \dots, x_n^*),$$

where  $1 < r < n$  and  $[x_1^*, \dots, x_n^*]$  is a permutation of  $[x_1, \dots, x_n]$ . This definition of functional separability is due to Pobarov.<sup>3)</sup> There is another definition by Shannon,<sup>4)</sup> but his definition has a less applicability than Pobarov's since he imposes the condition  $1 < r < n-1$  instead of  $1 < r < n$ . Thus, for example, a function of the form  $g(x_1, \dots, x_{n-1}) \oplus x_n$  is functionally separable for Pobarov's definition but it is not for Shannon's.

In dealing with functionally separable functions, it is often required to specify the variables  $x_1^*, \dots, x_r^*$  and the function  $g$ . For this reason, the author refines the definition as follows: The function  $f$  of the above definition is said to be *functionally separable with respect to*  $x_1^*, \dots, x_r^*$  and have the *kernel*  $g$ . Note here that a kernel is a non-degenerate function of its variables, and further that, if  $g$  is a kernel,  $g'$  is so too.

The notion of functional separability is of a practical importance in the synthesis of switching circuits, because functionally separable functions usually permit economical circuit realizations. Accordingly, it is desirable to find some criteria for this aspect of the structure of Boolean functions. One of such criteria is given by Pobarov, who stated it in the following theorem.

*Theorem 2.4.1. A function  $f(x_1, \dots, x_n)$  is functionally separable if and only if it is non-degenerate and there is a function  $g(x_1^*, \dots, x_r^*)$  such that all coefficients, other than 0 and 1, of the expansion of  $f$  with respect to variables  $x_{r+1}^*, \dots, x_n^*$  are equal to either  $g$  or  $g'$ , where  $1 < r < n$  and  $[x_1^*, \dots, x_n^*]$  is a permutation of  $[x_1, \dots, x_n]$ .*

*Proof.* Obvious from the definition.

The theorem is stated with a theoretical simplicity but it needs fairly large amount of calculations for its practical use. In fact, in order to determine whether a function is functionally separable or not, we must expand it with respect to every possible set of variables  $[x_{r+1}^*, \dots, x_n^*]$  and compare all coefficients other than 0 and 1. But, when we find that some coefficient other than 0 and 1 is a degenerate function of variables  $x_1^*, \dots, x_r^*$ , we may immediately conclude that the function is not functionally separable with respect to  $x_1^*, \dots, x_r^*$ .

<sup>1)</sup> See page 93. <sup>2)</sup> cf. Ref. 36. <sup>3)</sup> cf. Ref. 29. <sup>4)</sup> cf. Ref. 36.

and proceed to another expansion. Further, when the symmetry structure of the function is taken into account, the amount of the test procedures will somewhat be reduced, as will be seen below. At any rate, it is not, in general, an easy task to determine whether a function is functionally separable or not.

*Example 2.4.1.* Determine whether the function:

$$f(x, y, z) = x'y'z + x'yz + xy'z + xyz'$$

is functionally separable or not.

*Solution.* Expanding  $f$  with respect to  $x$ , we obtain  $f = x'z + x(y \oplus z)$ . But  $z \neq y \oplus z$  and  $z' \neq y \oplus z$ . Hence  $f$  is not functionally separable with respect to  $y$  and  $z$ .

Expanding  $f$  with respect to  $y$ , we obtain  $f = y'z + y(x \oplus z)$ . But  $z \neq x \oplus z$  and  $z' \neq x \oplus z$ . Hence  $f$  is not functionally separable with respect to  $x$  and  $z$ .

Expanding  $f$  with respect to  $z$ , we obtain  $f = z'xy + z(x' + y')$ . This time,  $(xy)' = x' + y'$ . Hence  $f$  is functionally separable with respect to  $x$  and  $y$  and can be written as  $f = xy \oplus z$ .

The functional separability or inseparability is obviously preserved under symmetries and complementation, i.e.,  $\sigma f$  for any symmetry  $\sigma$  and  $f'$  are functionally separable if and only if  $f$  is so. Thus, we have:

*Theorem 2.4.2.* Functions of the same genus are either all functionally separable or all not.

On the other hand, there is a close relationship between the functional separability and the symmetry structure of Boolean functions. The most apparent fact in this respect is that, when a function  $f$  is functionally separable and has the kernel  $g$ , the symmetry group of  $g$  is a subgroup of the symmetry group of  $f$ . Hence a function can be functionally separable only when its symmetry group contains some subgroup which is competent to be the symmetry group of another function of fewer variables. Thus, for example, no perfectly asymmetric function of four variables is functionally separable, since no function of less than four variables is perfectly asymmetric.

Now we are going to derive a series of theorems clarifying more complicated natures of the relationship. To begin with, concerning the functional separability of a function which is invariant under the symmetry of the form  $(ab)$ ,  $[ab](ab)$  or  $[c](ab)$ , we have:

*Theorem 2.4.3.* Let  $f$  be a function which is functionally separable with respect to a set of variables  $A$ , and let  $x_a$ ,  $x_b$  and  $x_c$  be any three variables such  $x_a \in A$ ,  $x_b \notin A$  and  $x_c \in A$ . Under this condition: (1) If  $f$  is invariant under  $(ab)$  or  $[ab](ab)$ , then  $f$  is functionally separable with respect to  $A \cup [x_b]$ ; (2)  $f$  is never invariant under  $[a](ab)$  and  $[b](ab)$ ; (3)  $f$  is never invariant under  $[c](ab)$ .

*Proof.* Expand  $f$  with respect to all variables outside  $A$  except  $x_b$  and choose an arbitrary coefficient of expansion  $h$  other than 0 and 1. Again expand  $h$  with respect to  $x_a$  and  $x_b$ , and write

$$h = x'_a x'_b h_0 + x'_a x_b h_1 + x_a x'_b h_2 + x_a x_b h_3.$$

Further, rewrite this as  $h = x'_b g_0 + x_b g_1$ ,

where  $g_0 = x'_a h_0 + x_a h_2$  and  $g_1 = x'_a h_1 + x_a h_3$ .

Now, since  $f$  is functionally separable with respect to  $A$ , and  $g_0$  and  $g_1$  are coefficients of expansion with respect to all variables outside  $A$ , each of them is equal to  $g$  or  $g'$  unless it is 0 or 1, where  $g$  is the kernel of  $f$ .

*Proof of (1):* It suffices to consider the case where  $f$  is invariant under  $(ab)$ ; for, if  $f$  is invariant under  $[ab](ab)$ ,  $[a]f$  is invariant under  $(ab)$ .

Now assume that  $f$  is invariant under  $(ab)$ . Then, since  $h$  is also invariant under  $(ab)$ , we have

$$(ab)h = x'_a x'_b h_0 + x'_a x_b h_2 + x_a x'_b h_1 + x_a x_b h_3 = h,$$

and consequently,  $h_1 = h_2$ .

It follows then that there are exactly three cases: (i)  $h_1 = h_2 = 0$ , (ii)  $h_1 = h_2 = 1$ , (iii)  $h_1 = h_2$  are neither 0 nor 1. Let us examine these cases in detail.

(i) When  $h_1 = h_2 = 0$ , we have  $g_0 = x'_a h_0$  and  $g_1 = x_a h_3$ . First, it is evident that  $g_0 \neq 1$ ,  $g_1 \neq 1$ ,  $g_0 \neq g_1$  unless  $g_0 = g_1 = 0$  ( $h_0 = h_3 = 0$ ) and  $g_0 \neq g'_1$  unless  $g_0 = x'_a$  and  $g_1 = x_a$  ( $h_0 = h_3 = 0$ ). But both provisos are unnecessary, since, if  $g_0 = g_1 = 0$ ,  $h$  is 0, while, if  $g_0 = x'_a$  and  $g_1 = x_a$ ,  $g$  is degenerate. Hence exactly either of  $g_0$  or  $g_1$  must be 0, since, otherwise, we have  $g_0 = g_1$  or  $g_0 = g'_1$ . It follows then that

either (a)  $g_0 = 0$  ( $h_0 = 0$ ),  $g_1 = x_a h_3$  and  $h = x_a x_b h_3$

or (b)  $g_0 = x'_a h_0$ ,  $g_1 = 0$  ( $h_3 = 0$ ) and  $h = x'_a x'_b h_0$ .

(ii) When  $h_1 = h_2 = 1$ , we have  $g_0 = x_a + h_0$  and  $g_1 = x'_a + h_3$ . First, it is evident that  $g_0 \neq 0$ ,  $g_1 \neq 0$ ,  $g_0 \neq g_1$  unless  $g_0 = g_1 = 1$  ( $h_0 = h_3 = 1$ ) and  $g_0 \neq g'_1$  unless  $g_0 = x_a$  and  $g_1 = x'_a$  ( $h_0 = h_3 = 0$ ). But both provisos are unnecessary, since, if  $g_0 = g_1 = 1$ ,  $h$  is 1, while, if  $g_0 = x_a$  and  $g_1 = x'_a$ ,  $g$  is degenerate. Hence exactly either of  $g_0$  or  $g_1$  must be 1, since, otherwise, we have  $g_0 = g_1$  or  $g_0 = g'_1$ . It follows then that

either (c)  $g_0 = 1$  ( $h_0 = 1$ ),  $g_1 = x'_a + h_3$  and  $h = x'_a + x'_b + h_3$

or (d)  $g_0 = x_a + h_0$ ,  $g_1 = 1$  ( $h_3 = 1$ ) and  $h = x_a + x_b + h_0$ .

(iii) When  $h_1 = h_2$  are neither 0 nor 1, both of  $g_0$  and  $g_1$  are neither 0 nor 1, and consequently, either  $g_0 = g_1$  or  $g_0 = g'_1$ . If we assume  $g_0 = g_1$ , we have  $h_0 = h_1 = h_2 = h_3$ , and therefore,  $g_0 = g_1 = h_0$ . But this is impossible because this means that  $g$  is independent of  $x_a$ . Hence  $g_0 = g'_1$ , i.e.,  $h'_0 = h_1 = h_2 = h'_3$ . It follows then that

$$(e) \quad g_0 = x_a \oplus h_0, \quad g_1 = (x_a \oplus h_0)' \quad \text{and} \quad h = x_a \oplus x_b \oplus h_0.$$

Thus, we have shown that there are five cases (a), (b), (c), (d) and (e) for

an arbitrary  $h$ . Now observe: The kernel  $g$  is  $x_a h_3$  or  $x'_a + h'_3$  for (a),  $x'_a h_0$  or  $x_a + h'_0$  for (b),  $x'_a + h_3$  or  $x_a h'_3$  for (c),  $x_a + h_0$  or  $x'_a h'_0$  for (d) and  $x_a \oplus h_0$  or  $x_a \oplus h'_0$  for (e). Accordingly, there are exactly three possibilities:

(I) either  $g_0 = 0$ ,  $g_1 = x_a \bar{h}$  and  $h = x_a x_b \bar{h}$

or  $g_0 = 1$ ,  $g_1 = x'_a + \bar{h}'$  and  $h = x'_a + x'_b + \bar{h}'$

for every  $h$ ;

(II) either  $g_0 = x'_a \bar{h}$ ,  $g_1 = 0$  and  $h = x'_a x'_b \bar{h}$

or  $g_0 = x_a + \bar{h}'$ ,  $g_1 = 1$  and  $h = x_a + x_b + \bar{h}'$

for every  $h$ ;

(III) either  $g_0 = x_a \oplus \bar{h}$ ,  $g_1 = x_a \oplus \bar{h}'$  and  $h = x_a \oplus x_b \oplus \bar{h}$

or  $g_0 = x_a \oplus \bar{h}'$ ,  $g_1 = x_a \oplus \bar{h}$  and  $h = x_a \oplus x_b \oplus \bar{h}$ ,

for every  $h$ ;

where  $\bar{h}$  is a non-degenerate function of all variables of  $A$  except  $x_a$ .

In each of the three possibilities, the forms of  $h$  indicate that  $f$  is functionally separable with respect to  $A \cup [x_b]$ , the kernel being  $x_a x_b \bar{h}$  or  $x'_a + x'_b + \bar{h}'$  for (I),  $x'_a x'_b \bar{h}$  or  $x_a + x_b + \bar{h}'$  for (II) and  $x_a \oplus x_b \oplus \bar{h}$  or  $x_a \oplus x_b \oplus \bar{h}'$  for (III).

*Proof of (2):* It suffices to prove that  $f$  is never invariant under  $[a](ab)$ , since, if  $f$  is invariant under  $[b](ab)$ ,  $[a]f$  is invariant under  $[a](ab)$ .

Now assume  $f$  be invariant under  $[a](ab)$ . Then, since  $h$  is invariant under  $[a](ab)$ , we have

$$[a](ab)h = x'_a x'_b h_1 + x'_a x_b h_3 + x_a x'_b h_0 + x_a x_b h_2 = h,$$

and consequently,

$$h_0 = h_1 = h_2 = h_3.$$

Thus, we arrive at a contradiction  $g_0 = g_1 = h_0$ , since this means that  $g$  is independent of  $x_a$ .

*Proof of (3):* Assume  $f$  be invariant under  $[c](ab)$ . Then, since  $h$  is also invariant under  $[c](ab)$ , we have

$$[c](ab)h = x'_a x'_b [c]h_0 + x'_a x_b [c]h_2 + x_a x'_b [c]h_1 + x_a x_b [c]h_3 = h,$$

and consequently,  $[c]h_0 = h_0$ ,  $[c]h_3 = h_3$  and  $[c]h_1 = h_2$ .

It follows from  $[c]h_1 = h_2$  that there is exactly three cases: (i)  $h_1 = h_2 = 0$ , (ii)  $h_1 = h_2 = 1$ , (iii)  $h_1 = h_2$  are neither 0 nor 1. When we examine these three cases, the five cases of (1) will be reproduced. But, in each of the five cases, we have a contradiction that  $g$  is independent of  $x_c$ , since  $h_0$  and  $h_3$  are independent of  $x_c$ . This completes the proof.

Next, as a generalization of (1) of Theorem 2.4.3, we have:

*Theorem 2.4.4.* Let  $A$  and  $B$  be any subsets of  $[x_1, \dots, x_n]$  such that  $A \cap B \neq \emptyset$  and  $A^c \cap B \neq \emptyset$ , then a function which is functionally separable with respect to  $A$  and symmetric with respect to  $B$  is functionally separable with respect to  $A \cup B$ ,  $B$  and  $A \cap B^c$ .

*Proof.* The proof can be carried out by induction.

First, assume that  $A^c \cup B$  consists of only one variable, say,  $x_b$ . Choose any variable  $x_a$  from  $A \cap B$ . Then  $f$  is invariant under  $(ab)$ , so, by (1) of Theorem 2.4.3,  $f$  is functionally separable with respect to  $A \cup [x_b] = A \cup B$ , the kernel  $h$  being  $x_a \bar{h}$  or  $x'_a + \bar{h}'$  for (I),  $x'_a \bar{h}$  or  $x_a + \bar{h}'$  for (II) and  $x_a \oplus \bar{h}$  for (III), where  $\bar{h}$  is a non-degenerate function of variables of  $A \cup B$  except  $x_a$ . Now we shall investigate the structure of  $h$  for each of the three possibilities, using the property that  $h$  is symmetric with respect to  $B$ .

(I) We may put  $h = x_a \bar{h}$ . Then we have  $h \leq x_a$ , and therefore, by symmetry,  $h \leq x_i$  for all the variables  $x_i$  of  $B$ . It follows that  $h \leq x_a x_b \cdots x_s$  where we put  $B = [x_a, x_b, \dots, x_s]$ . Thus, we conclude that

$$h = x_a x_b \cdots x_s h^* \text{ or } h = x'_a + x'_b + \cdots + x'_s + h^{*'},$$

where  $h^*$  is a non-degenerate function of the variables of  $A \cap B^c$ .

(II) We may put  $h = x'_a \bar{h}$ . Then, by the same reasoning as above, we arrive at the conclusion that

$$h = x'_a x'_b \cdots x'_s h^* \text{ or } h = x_a + x_b + \cdots + x_s + h^{*'}.$$

(III)  $h = x_a \oplus \bar{h}$ . Adding  $x_a \oplus x_b \oplus \cdots \oplus x_s$  (ring sum), we obtain

$$h \oplus x_a \oplus x_b \oplus \cdots \oplus x_s = \bar{h} \oplus x_b \oplus \cdots \oplus x_s.$$

The left hand side of the equation is obviously symmetric with respect to  $B$ . Further, it is independent of  $x_a$ , since the right hand side is so. Therefore, by symmetry, it is independent of all the variables of  $B$ . Thus, we conclude that  $h \oplus x_a \oplus x_b \oplus \cdots \oplus x_s = h^*$ , or equivalently,

$$h = x_a \oplus x_b \oplus \cdots \oplus x_s \oplus h^*.$$

Now observe: In each of the three possibilities,  $h$  is functionally separable with respect to both of  $B$  and  $A \cap B^c$ . Hence  $f$  is also functionally separable with respect to both of  $B$  and  $A \cap B^c$ .

Next, we shall prove the theorem when  $A^c \cap B$  consists of  $m+1$  variables under the assumption that the theorem is proved when  $A^c \cap B$  consists of  $m$  variables.

Let  $x_b$  be any variable of  $A^c \cap B$ , and let  $B^*$  be the set of all the variables of  $B$  except  $x_b$ . Obviously  $A^c \cap B^*$  consists of  $m$  variables, and  $A \cap B^* \neq \emptyset$ . Moreover,  $f$  is symmetric with respect to  $B^*$ . Hence, by the above assumption,  $f$  is functionally separable with respect to  $A^* = A \cup B^*$ . Here observe:  $A^* \cap B \neq \emptyset$  and  $A^* \cap B^c = [x_b]$ . Therefore  $f$  is functionally separable with respect to  $A^* \cup B = A \cup B$ ,  $B$  and  $A^* \cap B^c = A \cap B^c$ .

Thus, by induction, the theorem is proved.

The above two theorems are useful for reducing the test procedure of functional separability when the knowledge of symmetry structure is available. For example, when a function is known to be symmetric with respect to a set of variables  $A$ , it is sufficient to expand the function with respect to sets of variables  $B$  such that  $A \cap B = \emptyset$  or  $A \subset B$ .

Besides this usefulness, Theorem 2.4.4 has very interesting consequences, Theorem 2.4.5 and Theorem 2.4.6 which are originally due to Pobarov<sup>1)</sup>.

*Theorem 2.4.5. A function of  $n$  variables  $x_1, x_2, \dots, x_n$  which is symmetric as well as functionally separable is nothing but one of the six functions:  $x_1 x_2 \dots x_n$ ,  $x'_1 x'_2 \dots x'_n$ ,  $x_1 + x_2 + \dots + x_n$ ,  $x'_1 + x'_2 + \dots + x'_n$ ,  $x_1 \oplus x_2 \oplus \dots \oplus x_n$ ,  $(x_1 \oplus x_2 \oplus \dots \oplus x_n)'$ .*

*Proof.* Almost obvious by the proof of Theorem 2.4.4.

*Theorem 2.4.6. A function of  $n$  variables which is symmetrizable as well as functionally separable is nothing but an atom, an anti-atom or a linear function of the length  $n$ .*

*Proof.* Obvious by Theorem 2.4.2 and Theorem 2.4.5.

*Example 2.4.2.* Test the functional separability of the function:

$$f(w, x, y, z) = wxy + wxz + wyz + xyz + x'y'z'.$$

*Solution.* As will be seen easily,  $f$  is symmetric with respect to  $x, y$  and  $z$ . So we may only expand  $f$  with respect to  $w$ . Thus, we have

$$f = w'(xyz + x'y'z') + w(xy + xz + yz + x'y'z').$$

But  $w$ -coefficient is equal to neither  $w'$ -coefficient nor its complement. Hence  $f$  is not functionally separable.

## 2.5. Monotonous Boolean Functions

Let  $(\alpha_1, \alpha_2, \dots, \alpha_n)$  and  $(\beta_1, \beta_2, \dots, \beta_n)$  be any two ordered  $n$ -tuples of elements of  $B_0$ , then we define  $(\alpha_1, \alpha_2, \dots, \alpha_n) \leq (\beta_1, \beta_2, \dots, \beta_n)$  if and only if  $\alpha_i \leq \beta_i$  for every  $i$ . Evidently this relation is a partial order by which the class of all such  $n$ -tuples  $B^n$  becomes a Boolean algebra,

where  $0 = (0, 0, \dots, 0)$ ,

$1 = (1, 1, \dots, 1)$ ,

$(\alpha_1, \alpha_2, \dots, \alpha_n) + (\beta_1, \beta_2, \dots, \beta_n) = (\alpha_1 + \beta_1, \alpha_2 + \beta_2, \dots, \alpha_n + \beta_n)$ ,

$(\alpha_1, \alpha_2, \dots, \alpha_n) \cdot (\beta_1, \beta_2, \dots, \beta_n) = (\alpha_1 \beta_1, \alpha_2 \beta_2, \dots, \alpha_n \beta_n)$ ,

and  $(\alpha_1, \alpha_2, \dots, \alpha_n)' = (\alpha'_1, \alpha'_2, \dots, \alpha'_n)$ .

As we saw already in Theorem 2.1.1, a Boolean function  $f$  of  $n$  variables is completely specified by  $2^n$  values  $f(\alpha_1, \alpha_2, \dots, \alpha_n) \in B_0$ . Hence it can be regarded as a mapping from  $B^n$  into  $B_0$ .

<sup>1)</sup> cf. Ref. 29.



A function  $f$  which can be regarded as an order-preserving mapping from  $B^n$  into  $B_0$ , or in other words, a function  $f$  such that  $f(\alpha_1, \alpha_2, \dots, \alpha_n) \leq f(\beta_1, \beta_2, \dots, \beta_n)$  if  $(\alpha_1, \alpha_2, \dots, \alpha_n) \leq (\beta_1, \beta_2, \dots, \beta_n)$ , is said to be *monotonous*.

*Theorem 2.5.1. A function is monotonous if and only if it can be formed from the variables through additions and multiplications only (without use of complementations).*

*Proof. "If" part:* First note that, for any two monotonous functions, their sum and product are both monotonous. Accordingly, it suffices to show that every variable  $x_i$  is monotonous. But this is obvious since  $x_i(\alpha_1, \alpha_2, \dots, \alpha_n) = \alpha_i$ .

*"Only if" part:* For any monotonous function  $f$ , let  $S$  be the set of all  $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n)$  such that  $f(\alpha_1, \alpha_2, \dots, \alpha_n) = 1$ . Further, let  $\alpha^{(i)} = (\alpha_1^{(i)}, \alpha_2^{(i)}, \dots, \alpha_n^{(i)})$  be a minimal element of  $S$  and  $S_i$  be the subset of  $S$  formed by all  $\alpha \in S$  such that  $\alpha \geq \alpha^{(i)}$ . Then it is observed that  $S = \bigcup_i S_i$  where the union is taken for all minimal elements  $\alpha^{(i)}$ . Evidently  $f$  is the sum of atoms  $x_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n}$  such that  $(\alpha_1, \alpha_2, \dots, \alpha_n) \in S$ , and therefore, it is given by  $f = \sum_i f_i$  where  $f_i$  is the sum of atoms  $x_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n}$  such that  $(\alpha_1, \alpha_2, \dots, \alpha_n) \in S_i$ .

We shall now show that  $f_i$  is a product of variables. Without loss of generality, we may assume that  $\alpha_1^{(i)} = \alpha_2^{(i)} = \dots = \alpha_s^{(i)} = 1$  and  $\alpha_{s+1}^{(i)} = \dots = \alpha_n^{(i)} = 0$ . Then  $f_i$  is the sum of all the atoms  $x_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n}$  such that  $\alpha_1 = \alpha_2 = \dots = \alpha_s = 1$ , and hence, is given by:

$$\begin{aligned} f_i &= \sum x_1 x_2 \dots x_s x_{s+1}^{\alpha_{s+1}} \dots x_n^{\alpha_n} = x_1 x_2 \dots x_s \sum x_{s+1}^{\alpha_{s+1}} \dots x_n^{\alpha_n} \\ &= x_1 x_2 \dots x_s \prod_{j=s+1}^n (x_j + x_j^0) = x_1 x_2 \dots x_s. \end{aligned}$$

Thus, it has been shown that  $f$  is represented by a sum of products of variables. This completes the proof.

We see from the theorem that the class of all monotonous functions of  $n$  variables is nothing but a *free distributive lattice generated from  $n$  variables*. The problem of counting the number  $F_n$  of elements of free distributive lattice generated from  $n$  variables was proposed by Dedekind and has been solved up to  $n = 6$ . The results are reproduced in Table 2.5.1. Some estimates of  $F_n$  for larger values of  $n$  and further accounts related to monotonous functions was given by Gilbert<sup>1)</sup>.

Obviously, for any function  $f$  and any permutation operator  $\pi$ ,  $\pi f$  is monotonous if and only if  $f$  is monotonous. But, for any complementation operator  $\tau$ ,  $\tau f$  is not monotonous even if  $f$  is monotonous. Hence the monotony is not preserved under symmetries. However, it can be said in general that a function is congruent with a monotonous function if and only if it can be formed from  $n$  literals  $x_1^{\alpha_1}, x_2^{\alpha_2}, \dots, x_n^{\alpha_n}$  by additions and multiplications only.

TABLE 2.5.1

$n$	$F_n$
1	3
2	6
3	20
4	168
5	7,581
6	7,828,354

<sup>1)</sup> cf. Ref. 8.

## 2.6. Geometric Representations of Boolean Functions

### 2.6.1. Representation by $n$ -Cube.

Associate with each atom  $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$  a point of  $n$ -dimensional space with the coordinates  $(\alpha_1, \alpha_2, \dots, \alpha_n)$ . Thus, we obtain a one-to-one correspondence between atoms and vertices of the unit hypercube of  $n$ -dimensional space, or simply,  $n$ -cube. Since any Boolean function is represented uniquely by a sum of atoms, it is represented uniquely by a set of vertices of  $n$ -cube. For example, the functions of three variables  $f = \sum(0, 6, 7)$  and  $g = \sum(1, 2, 5)$  are represented by the sets of three vertices shown in Fig. 2.6.1 and Fig. 2.6.2 respectively.

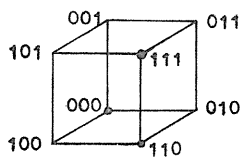


FIG. 2.6.1

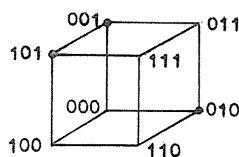


FIG. 2.6.2

At a glance, it will be recognized that  $f$  and  $g$  are congruent with each other. The relation between them is given by  $g = [2](13)f$ .

Now a *distance* is defined between vertices of  $n$ -cube as follows: For every pair of vertices  $p = (\alpha_1, \alpha_2, \dots, \alpha_n)$  and  $q = (\beta_1, \beta_2, \dots, \beta_n)$ , the distance between them is defined by:  $d(p, q) = \sum_{i=1}^n |\alpha_i - \beta_i|$ . As will be easily verified, the distance axioms are satisfied, that is,

- (1)  $d(p, q) = 0$ , if and only if  $p = q$ ,
- (2)  $d(p, q) \geq 0$  for every pair of vertices  $p$  and  $q$ ,
- (3)  $d(p, r) \leq d(p, q) + d(q, r)$  for any three vertices  $p, q$  and  $r$ .

In connection with this distance, a *sphere* can be defined on  $n$ -cube as follows: Let  $p_0$  be an arbitrary vertex, then, by a sphere of radius  $r$  about  $p_0$  is meant the set of vertices  $p$  such that  $d(p, p_0) = r$ . By this definition, every sphere has two centers which are called as the *near-center* and the *far-center*<sup>1)</sup>. As an illustration, consider the two spheres on 4-cube consisting of the following sets of vertices:

- |                      |                  |
|----------------------|------------------|
| (a) (0, 0, 0, 1) and | (b) (1, 1, 1, 0) |
| (0, 0, 1, 0)         | (1, 1, 0, 1)     |
| (0, 1, 0, 0)         | (1, 0, 1, 1)     |
| (1, 0, 0, 0)         | (0, 1, 0, 0)     |
|                      | (0, 0, 1, 0)     |
|                      | (0, 0, 0, 1)     |

(a) is a sphere of radius 1 about the near-center (0, 0, 0, 0) and at the same time a sphere of radius 3 about the far-center (1, 1, 1, 1). On the other hand,

<sup>1)</sup> cf. Ref. 15.

the near-center and the far-center of the sphere (b) are  $(1, 0, 0, 0)$  and  $(0, 1, 1, 1)$  but the radii are 2 for both centers. In general, if a vertex  $(\alpha_1, \alpha_2, \dots, \alpha_n)$  is the near-center of a sphere of radius  $r$ , then the vertex  $(\alpha'_1, \alpha'_2, \dots, \alpha'_n)$  is the far-center of the same sphere and the corresponding radius is  $n-r$  where we assume  $r \leq n-r$ .

Notice that a function is congruent with a fundamental symmetric function if and only if it is represented by a sphere. Accordingly, a function is symmetrizable if and only if it is represented by a set of cocentral spheres. Especially, a function is symmetric if and only if it is represented by a set of cocentral spheres about the centers  $(0, 0, \dots, 0)$  and  $(1, 1, \dots, 1)$ .

The notion of the distance has other important applications in the theory of error detecting and correcting codes but we shall not enter into such a problem.

### 2.6.2. Representation by $2^n$ -Cube

Associate with each Boolean function  $f$  of  $n$  variables a point  $(\beta_0, \beta_1, \dots, \beta_{2^n-1})$  in a  $2^n$ -dimensional space where we assume that  $f$  has the standard sum  $f = \sum \beta_i a_i$ . This gives rise to a one-to-one correspondence between Boolean functions of  $n$  variables and vertices of  $2^n$ -cube. Thus, in contrast with the representation by  $n$ -cube, every function is represented by a vertex of  $2^n$ -cube. In spite of this theoretical simplicity, this representation is of little practical use, because it requires a space of a very high dimension even for a moderate value of  $n$ . But it is of theoretical importance, opening up a way to the development of a theory of the coordinate representation which will be treated in detail in the next chapter.

### 2.6.3. Representation by the Karnaugh Map

As we saw already in Section 1.5, the Boolean algebra can be interpreted as the algebra of set calculus. Accordingly, Venn diagrams can be used to represent Boolean functions. In Fig. 2.6.3, a Venn diagram for the representation of Boolean functions of two variables is given. In the diagrams, each area represents an atom named by its entry. For example, the area which is inside the circle  $x$  and outside the circle  $y$  represents the atom  $xy'$ .

Venn diagrams using only circles can be used for as many as three variables, but beyond this, they are of no use, because we can not produce all possible areas corresponding to all atoms. Therefore some figures other than circles must be used for four or more than four variables. In Fig. 2.6.4 is given a Venn diagram for four variables using rectangles. It is noted that the 16 atoms are arranged in the form of a 4 by 4 matrix. The arrangement is made clearer when we simplify the diagram as shown in Fig. 2.6.5. The simplified diagram is a very excellent tool for manipulating Boolean functions of four variables and is called the *Karnaugh map*<sup>1)</sup>. In each cell of the map is entered the serial number of the atom represented by it. For example, the cell located at the intersection of the third row and the first column contains 14, because it lies within the regions

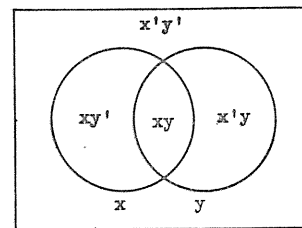


FIG. 2.6.3

<sup>1)</sup> cf. Ref. 13.

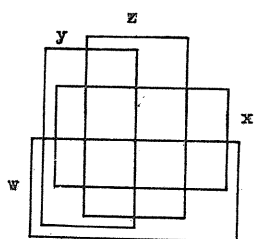


FIG. 2.6.4

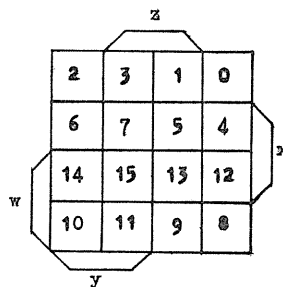


FIG. 2.6.5

$w$ ,  $x$  and  $y$  but outside the region  $z$  and therefore it represents the atom  $a_{14} = wxyz'$ .

An important feature of the map is that, if we start from any cell and move either horizontally or vertically, the adjacent cell we reach always represents an adjacent atom, where atoms are said to be *adjacent* if one is obtained from the other by complementing one of the variables, or equivalently, the distance between the corresponding vertices of  $n$ -cube is 1. These adjacencies exist not only within the interior of the map, but also between the two ends of each row and each column. That is: The bottom cell and the top cell in any column and the right cell and the left cell in any row are adjacent. In this sense, the Karnaugh map is to be regarded as a map drawn on the surface of a *torus*.

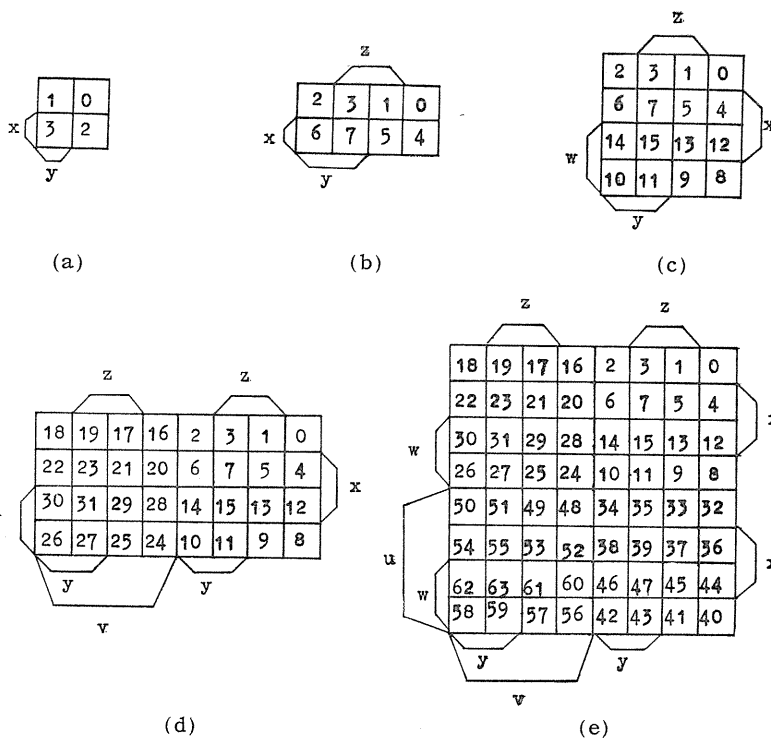


FIG. 2.6.6

Karnaugh maps are also formed for other numbers of variables. In Fig. 2.6.6, maps for two, three, five and six variables are shown and a map for four variables is reproduced for comparison. Maps for more than four variables are formed by combining 4-variable maps. For example, the map (d) for five variables is formed by combining two 4-variable maps; the map (e) for six variables is formed by combining four 4-variable maps. In maps for more than four variables, adjacencies exist not only within each 4-variable submap but also between any pair of corresponding cells of adjacent submaps. Thus, for example, in (e), not only the cells 16, 25, 26 and 28 but also the cells 8 and 56 are adjacent to the cell 24. In principle, Karnaugh maps can be formed for any number of variables but they are difficult to use if more than eight variables are involved.

Now we shall show how to plot a function on a Karnaugh map by an example.

*Example 2.6.1.* Plot the function of four variables:

$$f(w, x, y, z) = x'z' + wx'y' + wyz' + w'x'yz + w'xy'z' + w'xz$$

on a Karnaugh map.

*Solution.* One way to plot the function is to expand it into the standard sum and put a 1 in the cell representing each atom appearing in the standard sum. The standard sum of  $f$  is given by:

$$f(w, x, y, z) = \sum (0, 2, 3, 4, 5, 7, 8, 9, 10, 14),$$

so  $f$  is plotted as shown in (f) of Fig. 2.6.7.

However, one merit of the Karnaugh map is that such expansions are unnecessary. Fig. 2.6.7 shows the successive steps followed in plotting the function directly. The terms of the function are taken up one at a time, and 1's are put in the proper cells for each term.

First, we take up the term  $x'z'$ . We note that  $x'$  is represented by the top row together with the bottom row;  $z'$  is represented by the left column together with the right column;  $x'z'$  is thus represented by the cells common to  $x'$  and  $z'$ , i.e., the four corner cells. Next, we see that  $wx'y'$  lies in the right half to be in  $y'$ , in the bottom right quarter to be in both  $w$  and  $y'$ , and in the bottom right eighth to be in  $w$ ,  $x'$  and  $y'$ . Therefore we must put two 1's, one in 8 the other in 9. But 8 is already occupied, we may put only one 1 in 9. The remaining terms are plotted in the similar fashion. The completed map is shown in (f) and is the same as that obtained by the first method.

Thus, it is a relatively simple matter to plot a function given in an algebraic form. But the most important use of Karnaugh maps is to derive the simplest algebraic form of a function plotted on the map. The method for simplifying Boolean functions by Karnaugh maps will be explained in the next section together with another formal one. Here we add only that a familiarity with certain patterns of entries of the map called *subcubes* is indispensable for the simplification of Boolean functions as well as for other purposes. A *subcube* is defined as a set of cells of a map which represents a product of literals, because the counterpart on  $n$ -cube of such a set is a hypercube of a lower dimension. For example, in a 4-variable map, a set of two adjacent cells forms a subcube representing a product of three literals; a set of four cells each of which is adjacent to two of others in

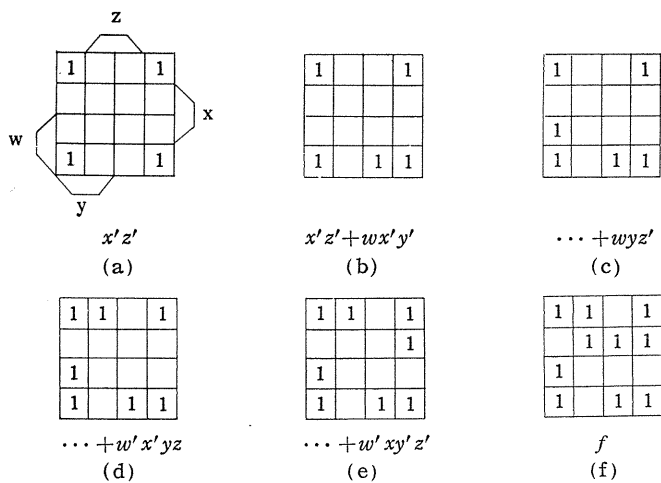


FIG. 2.6.7

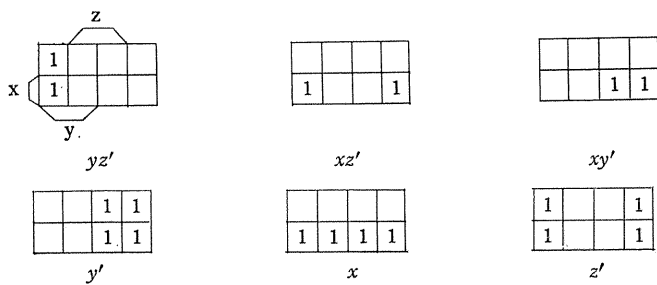


FIG. 2.6.8

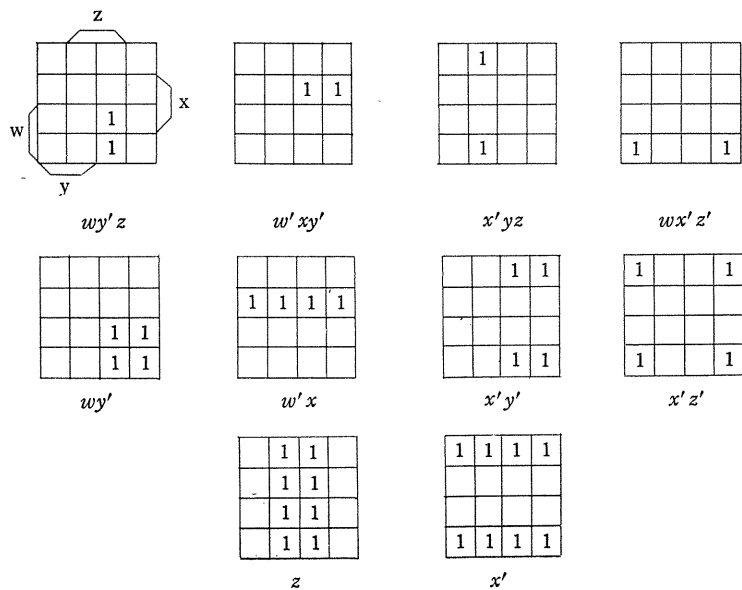


FIG. 2.6.9

the same set forms a subcube representing a product of two literals; a set of eight cells each of which is adjacent to three of others in the same set forms a subcube representing a single literal. In general, in an  $n$ -variable map, a set of  $2^p$  cells each of which is adjacent to  $p$  of others in the same set forms a subcube representing a product of  $n-p$  literals. In Fig. 2.6.8 and Fig. 2.6.9 below, typical subcubes in 3- and 4-variable maps are shown for the sake of illustration.

## 2.7. Simplification of Boolean Functions

Consider the function  $x+y'$ . The function can be written in many different forms:  $x+y' = xy+y' = x+x'y' = x'y'+xy'+xy$ . In this simple example already, we see the multiplicity of algebraic forms of Boolean functions. This opens up the possibility that some form may be the simplest of all and that there may be some method for arriving at such a form. However, any such method depends upon the standard of simplicity and many different standards are conceivable. In this section and throughout this paper, we are mainly interested in *normal sums* and *normal products*, i.e., sums of products and products of sums of literals. For these normal formulas, we define the minimal formulas as follows: A normal sum (product) is a *minimal sum* (*product*) if and only if it is *irredundant*, i.e., it cannot be simplified further by eliminating redundant literals or redundant products (sums) of literals, and it requires the minimum number of occurrences of literals<sup>1)</sup>. Thus, for example,  $x+y'$  is a minimal sum and a minimal product at the same time, while  $xy'+y'z+zx'$  is not a minimal sum because it is not irredundant and can be simplified as  $xy'+zx'$  by eliminating a redundant term  $y'z$ .

Here two facts may be stated in justification of the somewhat severe restriction to the normal formulas and the definition of the minimal formulas. One is that there actually exist some methods for arriving at these minimal formulas and the other is that there is a strong technical reason for the above restriction in applying Boolean functions to the synthesis of rectifier switching circuits. Especially the minimal sum is perhaps of primary importance for most cases. For this reason and because a minimal product of a function can be immediately obtained from a minimal sum of its complement by De Morgan's laws, we may concentrate our attention to the minimal sum in the remainder of this section.

### 2.7.1. Quine's Algorithm<sup>2)</sup>

We shall now explain *Quine's algorithm* for converting a normal sum into a minimal sum. To begin with, we must refine our terminology. A product of literals is called a *fundamental formula* if it contains no variable twice. A fundamental formula which is a summand of a normal sum  $f$  is called a *clause* of  $f$ . A fundamental formula  $\phi$  is said to *subsume* another fundamental formula  $\psi$ , if and only if the literals of  $\psi$  are among those of  $\phi$ . Thus, " $\phi$  subsumes  $\psi$ " is equivalent to " $\psi$  includes  $\phi$ ". A fundamental formula is a *prime implicant* of a function  $f$  if and only if it is included in  $f$  and subsumes no other fundamental formula included in  $f$ . Accordingly, if a clause  $\phi$  of a normal sum  $f$  is not a prime implicant of  $f$ , then there exists a prime implicant  $\psi$  of  $f$  subsumed by  $\phi$ . Since  $\phi$  can be replaced by  $\psi$  without altering the function  $f$ ,  $\phi$  contains redundant

<sup>1)</sup> The irredundancy does not imply the minimality. See Ref. 30.

<sup>2)</sup> cf. Ref. 30, 31 and 32.

literals. Therefore we have:

*Theorem 2.7.1. Any irredundant normal sum and, a fortiori, any minimal sum is a sum of prime implicants.*

Any function other than 0 and 1 has its prime implicants, because it includes at least one atom and, for each atom included in it, there is at least one prime implicant subsumed by the atom. Obviously the number of prime implicants of any function is finite.

*Theorem 2.7.2. Any function is equivalent to the sum of all its prime implicants.*

*Proof.* A function  $f$  includes each of its prime implicants, and hence, their sum. Conversely, every atom of  $f$  is included in one of its prime implicants. So their sum  $f$  is included in the sum of all its prime implicants.

In view of Theorem 2.7.1 and Theorem 2.7.2, a function can be converted into a minimal sum in two stages: (I) transform it into the sum of all its prime implicants, and then (II) delete from the sum the largest possible sum of jointly dispensable prime implicants. (I) can be carried out by a technique due to Samson and Mills<sup>1)</sup>, the explanation of which calls for one more definition. If two fundamental formulas  $\phi$  and  $\psi$  are opposed in exactly one variable, then they are said to have as their *consensus* the fundamental formula which we obtain from the formal product  $\phi\psi$  by deleting the two opposed literals and any repetitions of other literals.

*Theorem 2.7.3. Any normal sum can be eventually converted into the sum of all its prime implicants, if we apply the following two rules as far as possible.*

- (1) *If a clause subsumes another, drop the former.*
- (2) *Add, as an additional clause, the consensus of any two clauses, unless it does not subsume an existing clause.*

*Proof.* The theorem is proved by proving the following four lemmas:

- (a) *Normal sums go into normal sums under (1) and (2).*
- (b) *A normal sum remains susceptible to (2) as long as some prime implicant of it is not a clause of it.*
- (c) *A normal sum remains susceptible to (1) or (2) as long as some clause of it is not a prime implicant of it.*
- (d) *No normal sum is susceptible to (1) and (2) without end.*

*Proof of (a):* Obvious.

*Proof of (b):* Let  $f$  be a normal sum and  $\chi$  a prime implicant of  $f$  which is not a clause of  $f$ . Then there is at least one fundamental formula,  $\gamma$  anyway, which subsumes  $\chi$  and subsumes no clause of  $f$ . Let  $\phi$  be one of the longest fundamental formulas having the property. Now observe that  $\phi$  is not an atom; for, if it were, then, since  $\phi \leq \chi \leq f$ ,  $\phi$  would subsume some clause of  $f$ . But this contradicts to the property of  $\phi$ . So  $\phi$  lacks some variables of  $f$ , say,  $x_i$ . Since

<sup>1)</sup> cf. Ref. 34.



$\psi$  is a longest fundamental formula having the property, each of  $x_1\psi$  and  $x'_1\psi$ , evidently subsuming  $\chi$ , must subsume a clause of  $f$ . So there are clauses  $\phi_1$  and  $\phi_2$  of  $f$  subsumed respectively by  $x_1\psi$  and  $x'_1\psi$ . But  $\psi$  subsumes neither  $\phi_1$  nor  $\phi_2$ . Therefore  $x_1$  and  $x'_1$  must occur respectively in  $\phi_1$  and  $\phi_2$ . Further,  $\phi_1$  and  $\phi_2$  are not just  $x_1$  and  $x'_1$ , otherwise  $f$  would be 1: nor are they opposed in variables other than  $x_1$ , since their further literals are common to  $\psi$ . So  $\phi_1$  and  $\phi_2$  have a consensus, say,  $\phi$ . Moreover  $\phi$  subsumes no clause of  $f$ , for  $\phi$  is subsumed by  $\psi$  subsuming no clause of  $f$ . Hence  $f$  is susceptible to (2).

*Proof of (c):* Let  $f$  be a normal sum and  $\phi$  any clause of  $f$  which is not a prime implicant. Then,  $\phi$ , being included in  $f$ , subsumes a prime implicant  $\psi$ . If  $\psi$  is a clause of  $f$ , we can apply (1) to drop  $\phi$ , and otherwise we can apply (2) by (b).

*Proof of (d):* By the proviso of the rule (2), no clause once dropped by (1) can ever be restored by (2); neither can a clause already present be reintroduced by (2). Summing up, (2) can never introduce the same clause twice. But there are only finitely many different fundamental formulas for (2) to introduce. So the use of (2) must terminate eventually. Also the use of (1), being subtractive, must terminate eventually. This completes the proof of the theorem.

A standard sum is obviously a normal sum, so the technique can also be applied to a standard sum. Since Boolean functions are usually given by or easily turned into standard sum, it is worth while to describe the process of conversion of a standard sum into a sum of all prime implicants.

Now assume we are given a standard sum  $f$ , then the process will be accomplished in at most  $n-1$  steps as follows.

*First Step:* Compare any atom with all others, and whenever we find any two atoms which are opposed in exactly one variable, add their consensus to  $f$ , and put a mark on each of them. When all the possibilities are exhausted, drop every atom having at least one mark.

*Second Step:* Compare any clause added in the first step with all other similar clauses, and whenever we find any two clauses which contain the same set of variables and are opposed in exactly one variable, add their consensus to  $f$  and put a mark on each of them. When all the possibilities are exhausted, drop every clause having at least one mark.

.....

*k th Step:* Compare any clause added in the  $k-1$ -th step with all other similar ones, and whenever we find any two clauses which contain the same set of variables and are opposed in exactly one variable, add their consensus to  $f$  and put a mark on each of them. When all the possibilities are exhausted, drop every clause having at least one mark.

.....

It is observed that every clause which is added in  $k$ -th step is a product of  $n-k$  literals and, indeed, all the products of  $n-k$  literals included in  $f$  are added in the step. Accordingly, clauses added in  $n-1$ -th step, if any, are single literals,

and hence, the process must terminate at the latest in  $n-1$ -th step.

At the beginning of every  $k$ -th step, the normal sum  $f$  is not susceptible to the rule (1); for, otherwise, there would be a clause which is a product of more than  $n-k$  literals and subsumes another shorter clauses; but this is impossible because such a clause must have been dropped in a previous step. By the same reason, it is not necessary to add a consensus of any two clauses other than those stated above, because such a consensus is a product of more than  $n-k$  literals subsuming other clauses, its parent clauses anyway.

McCluskey<sup>1)</sup> recommends to count the number of unprimed variables of fundamental formulas—he calls it the index—and classify, preparatorily to each step, the clauses added in the preceding step into groups according to their indices. When this is done, it is sufficient for every clause in the group of the index  $i$  to be compared with those in the group of the index  $i+1$ , starting with the group of the lowest index and proceeding to the higher one consecutively. Thus, the amount of comparisons will be reduced considerably.

It is also recommended to pay attention to the number of marks put on clauses during the process of comparison; for, if a clause containing  $k$  literals has obtained  $k$  marks, it can be excluded from consideration thereafter.

So much for the stage (I), the conversion of a normal sum into the sum of all its prime implicants.

Now the stage (II) will be explained. A method for carrying out the stage is as follows.

*Step 1:* Construct a table having as many rows as prime implicants and as many columns as the atoms included in the given normal sum  $f$ . Identify each row with a prime implicant and each column with an atom. Enter a mark in a position of the table whenever the atom of the column subsumes the prime implicant of the row. This table is called the *table of prime implicants*. In terms of the table, the task (II) is to select the simplest combination or combinations of rows such that any column contains at least one mark on some of them.

*Step 2:* Examine the columns of the table. If a column contains only one mark, the corresponding prime implicant is obviously bound to appear in every irredundant normal sum and therefore in every minimal sum of  $f$ . Any of such prime implicants is said to be *essential* and the sum of all of them to be the *core*. Encircle the essential prime implicants together with all marks contained in the corresponding rows. Reduce the table by deleting all essential rows and all columns containing at least one encircled mark. Atoms, thus dropped, need not further be taken care of because they are already accounted for by essential prime implicants.

*Step 3:* Further examine the columns of the table. Whenever we find two columns such that one of them contains a mark on every row where the other contains a mark, delete the former, because every selection of rows accounting for the latter accounts for the former.

*Step 4:* Examine the rows of the table. If there are rows which remain but

---

<sup>1)</sup> cf. Ref. 21.

contain no mark, delete all such rows. Any of the prime implicants thus deleted appears in no minimal sum and is said to be *absolutely superfluous*. Clearly, a sufficient condition that a prime implicant may be absolutely superfluous is that it is not essential and, at the same time, it is included in the core. Quine<sup>1)</sup> conjectured that this might be also necessary. But it is found that the conjecture is wrong. An example proving the negative answer to the conjecture will be given as Example 2.7.3. below.

*Step 5:* Now examine the reduced table, and select the simplest set or sets of prime implicants which taken together include at least one mark in each column. The sum of the prime implicants together with the essential ones obtained in Step 2 forms a minimal sum of  $f$ . This step is simple or complicated as the case may be.

We shall now explain a systematic method for carrying out Step 5<sup>2)</sup>. Represent all prime implicants by letters, say,  $A, B, C$  etc. Form, for each atom, the sum of letters representing the prime implicants subsumed by it, so that, for example, when an atom subsumes prime implicants  $A, B$  and nothing else, we form the sum  $A+B$ . Multiply all such sums formally and construct an expression  $\emptyset$ . We shall now regard  $\emptyset$  as a Boolean function of the variables  $A, B, C$ , etc. If it is multiplied out and turned into a normal sum, every clause of it represents a set of prime implicants whose sum is equivalent to  $f$  and, indeed, every such set is represented by a clause. Further observe: If a clause of  $\emptyset$  subsumes another, it is redundant and may be dropped, because the corresponding sum of prime implicants is not an irredundant normal sum. Thus, when the formal product  $\emptyset$  is multiplied out and converted into an irredundant normal sum by deleting all redundant clauses, the remaining clauses represent all possible irredundant normal sums of  $f$ . Here, it is noted that the only irredundant normal sum, and consequently, the only minimal sum of  $\emptyset$  is the sum of all prime implicants of  $\emptyset$ . For, if an irredundant normal sum of  $\emptyset$  lacks a prime implicant, it should be susceptible to the rule (2); but this is impossible, since it contains no pair of clauses having a consensus. Hence we may convert  $\emptyset$  into the sum of all its prime implicants. A reasonable way to do this is to simplify the product  $\emptyset$  as far as possible before multiplying it out. That is: To begin with, delete every factor which is a single letter and every factor containing at least one of such letters. Any of these letters should be restored as a common factor of all prime implicants after the rest of  $\emptyset$  is converted into the sum of all its prime implicants. Obviously these letters represent essential prime implicants of  $f$  and this reduction is the same as that of Step 2 of the preceding table method.

Next, examine the rest of  $\emptyset$ , and whenever we find two factors such that one contains only letters contained in another, delete the latter. Of course, this reduction corresponds to that of Step 3 together with that of Step 4 of the table method.

The preparatory reduction of  $\emptyset$  having been completed, the rest of  $\emptyset$  is to be multiplied out and converted into a normal sum, which will be easily converted into the sum of all prime implicants by deleting all redundant clauses, i.e., clauses subsuming other clauses.

<sup>1)</sup> cf. Ref. 32. <sup>2)</sup> cf. Ref. 26.

When  $\emptyset$  is converted into the sum of all its prime implicants, it is a simple matter to obtain all minimal sums of  $f$ , because we may only count and compare the numbers of occurrences of literals for all irredundant normal sums of  $f$ .

In view of the parallelness between the reduction of the table of prime implicants and the preparatory reduction of  $\emptyset$ , it is also possible to start with the table of prime implicants, reduce the table and then apply the present method.

Now we shall work out a few examples.

*Example 2.7.1.* Simplify the four-variable function:

$$f(w, x, y, z) = \sum(0, 1, 2, 5, 7, 10, 11, 15).$$

*Solution.* The stage (I) is carried out as shown in Table 2.7.1.

The stage (II): The table of prime implicants is shown in Table 2.7.2.

TABLE 2.7.1

$wxyz$	$wxyz$
0** 0000	0 1 000-
1** 0001	0 2 00-0
2** 0010	1 5 0-01
5** 0101	2 10 -010
10** 1010	5 7 01-1
7** 0111	10 11 101-
11** 1011	7 15 -111
15** 1111	11 15 1-11

TABLE 2.7.2

	0 1 2 5 10 7 11 15
$A \ w'x'y'$	* *
$B \ w'x'z'$	* *
$C \ w'y'z'$	* *
$D \ x'yz'$	* *
$E \ w'xz$	* *
$F \ wx'y$	* *
$G \ xyz$	* *
$H \ wyz$	* *

Since there is no column containing only one check mark, no reduction of the table is possible. Now, representing the prime implicants by the letters given in the left column, we form the product:

$$\emptyset = (A+B)(A+C)(B+D)(C+E)(D+F)(E+G)(F+H)(G+H).$$

This can be transformed as follows:

$$\begin{aligned} \emptyset &= (A+B)(A+C) \cdot (F+H)(G+H) \cdot (B+D)(D+F) \cdot (C+E)(E+G) \\ &= (A+BC)(H+FG) \cdot (D+BF)(E+CG) \\ &= (AH+AFG+BCH+BCFG)(DE+BEF+CDG+BCFG) \\ &= (AH+AFG+BCH)(DE+BEF+CDE) + BCFG. \end{aligned}$$

When we multiply out the first term of the last expression, we obtain:

$$\begin{aligned} \emptyset &= ADEH + BCFG + ABEFG + ABEFH + ACDFG + ACDGH \\ &\quad + ADEFG + BCDEH + BCDGH + BCEFH. \end{aligned}$$

This normal sum being irredundant, we conclude that there are ten irredundant normal sums of  $f$  among which the two represented by the clauses  $ADEH$  and

*BCFG* are minimal. Thus, the minimal sums of  $f$  are given by:

$$f = w'x'y' + x'yz' + w'xz + wyz$$

and

$$f = w'x'z' + w'y'z + wx'y + xyz.$$

*Example 2.7.2.* Simplify the four-variable function:

$$f(w, x, y, z) = \sum(1, 3, 4, 6, 7, 13, 15).$$

*Solution.* The stage (I) is carried out as shown in Table 2.7.3.

The stage (II): The table of prime implicants is shown in Table 2.7.4. Examining the table, we find that the prime implicants  $w'x'z$ ,  $w'xz'$  and  $wxz$  are essential. Essential prime implicants are underlined and check marks on the corresponding rows are encircled. When the encircled rows and columns containing at least one encircled check mark are deleted, the table will be reduced to Table 2.7.5. This table is very simple. It shows that any of the remaining three prime implicants can be taken as a clause of a minimal sum of  $f$ . Hence we have three minimal sums:

$$f = w'x'z + w'xz' + wxz + \begin{cases} w'yz \\ w'xy \\ xyz. \end{cases}$$

TABLE 2.7.3

	$wxyz$		$wxyz$
1*	0001	1 3	00-1
4*	0100	4 6	01-0
3**	0011	3 7	0-11
6**	0110	6 7	011-
7***	0111	7 15	-111
13*	1101	13 15	11-1
15**	1111		

TABLE 2.7.4

	1	4	3	6	7	13	15
<u><math>w'x'z</math></u>	(*)		(*)				
<u><math>w'xz'</math></u>			(*)	(*)			
$w'yz$			*		*		
$w'xy$				*	*		
$xyz$					*		*
<u><math>wxz</math></u>						(*)	(*)

TABLE 2.7.5

	7
$w'yz$	*
$w'xy$	*
$xyz$	*

*Example 2.7.3.* Simplify the five-variable function:

$$f(v, w, x, y, z) = \sum(7, 15, 16, 17, 18, 20, 21, 22, 25, 26, 27, 29, 30, 31).$$

*Solution.* The stage (I) is shown in Table 2.7.6.

The stage (II): The table of prime implicants is shown in Table 2.7.7. There is only one essential prime implicant  $v'xyz$ . Thus, the top row and the columns

TABLE 2.7.6

$wxyz$		$wxyz$		$wxyz$	
16***	10000	16 17*	1000-	16 17 20 21	10-0-
17***	10001	16 18*	100-0	16 18 20 22	10--0
18***	10010	16 20**	10-00	17 21 25 29	1--01
20***	10100	17 21**	10-01	18 22 26 30	1--10
7*	00111	17 25*	1-001	25 27 29 31	11--1
21***	10101	18 22**	10-10	26 27 30 31	11-1-
22***	10110	18 26*	1-010		
25***	11001	20 21*	1010-		
26***	11010	20 22*	101-0		
15**	01111	7 15	0-111		
27***	11011	21 29*	1-101		
29***	11101	22 30*	1-110		
30***	11110	25 27*	110-1		
31****	11111	25 29**	11-01		
		26 27*	1101-		
		26 30**	11-10		
		15 31	-1111		
		27 31**	11-11		
		29 31*	111-1		
		30 31*	1111-		

7 and 15 are deleted. Then it is seen that the columns 20, 21, 22, 29, 30 and 31 can be deleted in favor of the columns 16, 17, 18, 25, 26 and 27 respectively. The resulting table is shown in Table 2.7.8. In the table, the top row contains no check mark and can be deleted. The prime implicant  $wxyz$  is thus found to be absolutely superfluous but it is not included in the core  $v'xyz$ . Hence it has been shown that an absolutely superfluous prime implicant is not necessarily included in the core.

Now representing the remaining six prime implicants by the letters given in the left column, we obtain:

TABLE 2.7.7

	16	17	18	20	7	21	22	25	26	15	27	29	30	31
$v'xyz$					(*)					(*)				
$wxyz$										*				*
$vw'y'$	*	*		*		*								*
$vw'z'$	*	*	*			*								
$vy'z$		*				*		*				*		
$vyz'$			*				*	*					*	
$vwz$							*		*		*	*	*	*
$vwy$								*		*	*	*	*	*

TABLE 2.7.8

	16	17	18	25	26	27
$wxyz$						
A $vw'y'$	*	*				
B $vw'z'$	*		*			
C $vy'z$		*		*		
D $vyz'$			*		*	
E $vwz$				*	*	
F $vwy$					*	*

$$\Phi = (A + B)(A + C)(B + D)(C + E)(D + F)(E + F),$$

which, when multiplied out and simplified, yields the minimal sum:

$$\Phi = ADE + BCF + ABEF + ADCF + BCDE.$$

Thus, restoring the core  $v'xyz$ , we obtain two minimal sums:

$$f = v'xyz + vw'y' + vyz' + vwz$$

and

$$f = v'xyz + vw'z' + vy'z + vwy.$$

As shown in the above examples, the minimal sums are not unique in general. Then it is natural to ask: "Are there any properties common to all the minimal sums of the same function and what properties?" We shall show one of such properties. To begin with, we define: For a function  $f$ , a literal is said to be either *essential* or *inessential* according as whether it appears in an irredundant normal sum of  $f$  or not.

*Lemma 2.7.1. Let  $x$  be a variable of a Boolean function  $f$ , and let  $f = xg + x'h$  be the expansion with respect to  $x$ , then:*

- (1)  $g = h$  if and only if  $x$  and  $x'$  are inessential,
- (2)  $g > h$  if and only if  $x$  is essential and  $x'$  is inessential,
- (3)  $g < h$  if and only if  $x$  is inessential and  $x'$  is essential,
- (4)  $g$  and  $h$  are incomparable, i.e., neither  $g > h$  nor  $g \leq h$  if and only if  $x$  and  $x'$  are essential.

*Proof.* It suffices to prove only the "if" parts because the four cases are mutually exclusive and exhaust all possibilities.

*Proof of (1):* If  $x$  and  $x'$  are inessential, then there is an irredundant normal sum of  $f$  where  $x$  and  $x'$  are absent. Hence  $f$  is independent of the variable  $x$ . It follows that  $f$  is expanded as  $f = xf + x'f$ , whence, by the uniqueness of the expansion, we have  $g = h = f$ .

*Proof of (2):* If  $x$  is essential and  $x'$  is inessential, then there is an irredundant normal sum of  $f$  where  $x$  is present but  $x'$  is absent. Summing up the clauses containing  $x$  and the remaining clauses separately,  $f$  can be written in the form:  $f = xg^* + h^*$  where  $g^*$  and  $h^*$  are independent of the variable  $x$ . Further, since  $h^* = (x + x')h^* = xh^* + x'h^*$ ,  $f$  can be turned into;  $f = xg + x'h$  where  $g = g^* + h^*$  and  $h = h^*$ , whence  $g \geq h$  is obvious. But  $g \neq h$ , for  $g = h$  implies  $g^* \leq h^*$  and, a fortiori,  $xg^* \leq h^*$ , so  $xg^*$ , the sum of clauses containing  $x$ , becomes redundant.

*Proof of (3):* Obvious from (2) by symmetry.

*Proof of (4):* If  $x$  and  $x'$  are essential, then there is an irredundant normal sum of  $f$  containing both  $x$  and  $x'$ . Summing up the clauses containing  $x$ , those containing  $x'$  and the rest separately, we obtain:  $f = xg^* + x'h^* + f^*$  where  $g^*$ ,  $h^*$  and  $f^*$  are independent of the variable  $x$ . Further, since  $f^* = xf^* + x'f^*$ ,  $f$  can be

rewritten as:  $f = xg + x'h$  where  $g = g^* + f^*$  and  $h = h^* + f^*$ . Now assume  $g \geq h$ , then it follows that  $g^* + f^* = g^* + h^* + f^*$ . Therefore  $f$  is turned into:  $f = xg^* + h^* + f^*$  which implies that all occurrences of the literal  $x'$  are redundant. Hence  $g \geq h$  and, by symmetry,  $g \leq h$  are impossible. This completes the proof.

Now, in view of the lemma, the following theorem is almost evident.

*Theorem 2.7.4. For any function  $f$ , any essential literal and no inessential literal appear in any irredundant normal sum (minimal sum) of  $f$ .*

*Proof.* We shall prove only the case where literals  $x$  and  $x'$  are essential for a function  $f$ . Other cases can be proved similarly.

Let  $f = xg + x'h$  be the expansion with respect to the variable  $x$ , then, from (4) of Lemma 2.7.1,  $g$  and  $h$  are incomparable. Now assume that there be an irredundant normal sum of  $f$  where at least one of  $x$  and  $x'$  is absent. Then it follows from (1), (2) or (3) of the lemma that  $g$  and  $h$  are comparable. But this is impossible in the light of the uniqueness of the expansion.

### 2.7.2. Simplification by the Karnaugh Map

We shall now explain how to simplify Boolean functions by means of Karnaugh maps. The process proceeds as follows:

*Stage (I).* Plot the given function on a Karnaugh map. Choose an arbitrary 1 cell. Examine the set of all subcubes containing the cell and find subcubes which are not contained in any other one of the same set. These subcubes obviously represent prime implicants subsumed by the atom represented by the cell. Repeat the same procedure for every 1 cell of the map. If a 1 cell has only one prime implicant, the prime implicant is essential. Whenever we find an essential prime implicant, put a dot in the corner of each 1 cell which is included in it. In the search for further prime implicants, we may omit the above procedure for every 1 cell which already contains a dot. Thus, it is advantageous first to try to find 1 cells which give rise to essential prime implicants, especially those containing only a few literals, because such prime implicants occupy a large number of 1 cells and consequently we can reduce subsequent work considerably.

*Stage (II).* When all the prime implicants are found, we must still choose the simplest set or sets of prime implicants whose sum is equivalent to the function. One way to do this may be to apply the methods described in Section 2.7.1, but under most circumstances the task can be carried out immediately by the inspection of the Karnaugh map.

Now, for the purpose of illustration, the three examples of Section 2.7.1 will be reattacked by the present method.

*Example 2.7.4.* Simplify the four-variable function of Example 2.7.1:

$$f(w, x, y, z) = \sum(0, 1, 2, 5, 7, 10, 11, 15).$$

*Solution.*  $f$  is plotted on a Karnaugh map as shown in Fig. 2.7.1. Examining the map, it will be readily seen that each 1 cell has two prime implicants and there are two simplest choices of prime implicants. These choices are given



in Fig. 2.7.2. Thus, we obtain the minimal sum:

$$f = w'x'y' + x'yz' + w'xz + wyz$$

from (a), and the minimal sum:

$$f = w'x'z' + w'y'z + wx'y + xyz$$

from (b). Of course, these results agree with those of Example 2.7.1.

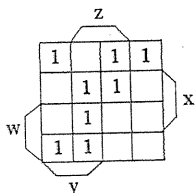
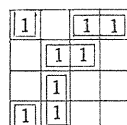
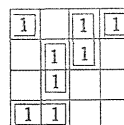


FIG. 2.7.1



(a)



(b)

FIG. 2.7.2

*Example 2.7.5.* Simplify the four-variable function of Example 2.7.2:

$$f(w, x, y, z) = \sum(1, 3, 4, 6, 7, 13, 15).$$

*Solution.*  $f$  is plotted on a Karnaugh map as shown in Fig. 2.7.3. Inspecting the map, it will be immediately found that there are three essential prime implicants  $w'x'z$ ,  $w'xz'$  and  $wxz$ , and the only one 1 cell which is free from a dot is included in three equally simple prime implicants  $w'yz$ ,  $w'xy$  and  $xyz$ . Hence we obtain three minimal sums:

$$f = w'x'z + w'xz' + wxz + \begin{cases} w'yz \\ w'xy \\ xyz. \end{cases}$$

*Example 2.7.6.* Simplify the five-variable function of Example 2.7.3:

$$\begin{aligned} f(v, w, x, y, z) \\ = \sum(7, 15, 16, 17, 18, 20, 21, 22, 25, 26, 27, 29, 30, 31). \end{aligned}$$

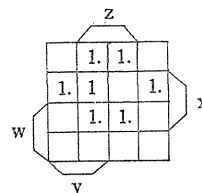


FIG. 2.7.3

*Solution.* The Karnaugh map of  $f$  is shown in Fig. 2.7.4. Examining the map, we find eight prime implicants. They are  $v'xyz$ ,  $vwv$ ,  $vw'y'$ ,  $vwz$ ,  $vw'z'$ ,  $vyz'$ ,  $vy'z$  and  $wxyz$  of which only  $v'xyz$  is essential. Now, observe that, in order to include the cell 27 (with \*), we must take either  $vwv$  or  $vwz$  as a clause of a minimal sum. In any case, the cell 31 (with \*) will be automatically included. Further observe that the cell 15 (with \*) has already gained a dot. Therefore the prime implicant  $wxyz$  may be discarded as absolutely superfluous. Then it will be seen that there are two simplest ways to cover all 1 cells which are free from dots. They are shown in Fig. 2.7.5. Thus, restoring the essential prime implicant  $v'xyz$ , we obtain two minimal sums:

$$f = v'xyz + vw'y' + vyz' + vwz$$

and

$$f = v'xyz + vw'z' + vy'z + vwy.$$

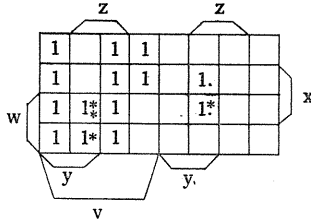
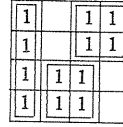
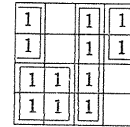


FIG. 2.7.4



(a)



(b)

FIG. 2.7.5

### 3. Theory of Coordinate Representation of Boolean Functions

In this chapter, a theory of coordinate representation of Boolean functions will be developed in a general manner. As will be seen, this representation seems rather awkward for practical manipulation of Boolean functions, but it possesses a peculiar adaptability for the study of structures of Boolean functions. The representation was initiated by D. E. Muller<sup>1)</sup>. The author owes the motivation for the present work to him, but the development in this and the next chapter were carried out independently.

#### 3.1. Boolean Matrix Representation of Boolean Functions

As we saw already,  $B_n$  is a Boolean ring with the unit 1 with respect to the ring addition and the multiplication where the addition and the complementation are expressed as  $f + g = f \oplus g \oplus fg$  and  $f' = 1 \oplus f$ .

Here we take another view point and regard any  $f \in B_n$  as an *operator*  $F$  transforming any  $h \in B_n$  into  $Fh = fh \in B_n$ . This gives rise to a one-to-one correspondence  $f \leftrightarrow F$  between Boolean functions  $f$  and operations  $F$ , since different functions correspond to different operators. Now let us define the *ring sum* and the *product* of operators by  $(FG)h = F(Gh)$  and  $(F \oplus G)h = Fh \oplus Gh$  respectively. Then, since  $(FG)h = F(Gh) = F(gh) = fgh$  and  $(F \oplus G)h = Fh \oplus Gh = fh \oplus gh = (f \oplus g)h$ , we see  $fg \leftrightarrow FG$  and  $f \oplus g \leftrightarrow F \oplus G$ . It follows that  $B_n$  can be represented by an *operator ring* isomorphic with itself. Of course, the *zero* 0, the *unity*  $I$ , the *sum*  $F + G$  and the *complement*  $F'$  of operators should be defined by  $0 \leftrightarrow 0$ ,  $1 \leftrightarrow I$ ,  $f + g \leftrightarrow F + G$  and  $f' \leftrightarrow F'$ , whence  $F + G = F \oplus G \oplus FG$  and  $F' = I \oplus F$ .

In Theorem 2.1.5, we found further that any Boolean function  $h \in B_n$  is represented uniquely by a ring sum of atoms as  $h = \sum_i \gamma_i a_i$  where  $\gamma_i \in B_0$  ( $i = 0, 1, \dots, 2^n - 1$ ). Accordingly, operators  $F$  are represented by  $2^n \times 2^n$  matrices on  $B_0$ , and the class of all such matrices is a ring isomorphic with  $B_n$ , where *matrix ring sum* and *matrix product* are defined by  $(\alpha_{ik}) \oplus (\beta_{ik}) = (\alpha_{ik} \oplus \beta_{ik})$  and

<sup>1)</sup> cf. Ref. 20.

$(\alpha_{ik})(\beta_{ik}) = (\sum_j^{\oplus} \alpha_{ij} \beta_{jk})$  respectively.

New let us determine the matrices  $F$  explicitly. Putting  $f = \sum_i^{\oplus} \alpha_i a_i$ , we have :  
 $Fh = fh = (\sum_i^{\oplus} \alpha_i a_i)(\sum_j^{\oplus} \gamma_j a_j) = \sum_i^{\oplus} \alpha_i \gamma_i a_i$ , whence we obtain  $F = (\alpha_i \delta_{ik})$  where  $\delta_{ik}$  is 1 if  $i = k$  and is 0 if  $i \neq k$ . Thus, we have :

**Theorem 3.1.1.** Associate with each Boolean function  $f = \sum_i^{\oplus} \alpha_i a_i$  a matrix  $(\alpha_i \delta_{ik})$ , then  $B_n$  is represented isomorphically by the class of all  $2^n \times 2^n$  diagonal matrices on  $B_0$ , where, if  $f \leftrightarrow (\alpha_i \delta_{ik})$  and  $g \leftrightarrow (\beta_i \delta_{ik})$ . (1)  $f + g \leftrightarrow ((\alpha_i + \beta_i) \delta_{ik})$ , (2)  $fg \leftrightarrow (\alpha_i \beta_i \delta_{ik})$ , (3)  $f \oplus g \leftrightarrow ((\alpha_i \oplus \beta_i) \delta_{ik})$ , (4)  $f' \leftrightarrow (\alpha'_i \delta_{ik})$ , (5)  $0 \leftrightarrow (0)$ ,  $1 \leftrightarrow (\delta_{ik})$ .

### 3.2. Real Matrix Representation of Boolean Functions

In the two-element Boolean algebra  $B_0$ , the elements 0 and 1 are not real numbers but merely two symbols. However, the behavior of the symbols under Boolean operations is very alike to that of real numbers 0 and 1 under corresponding ordinary operations. In order to see the resemblance, let us denote the real counterpart of an element  $\alpha$  of  $B_0$  by  $R(\alpha)$ . Then the following lemma will be obtained.

**Lemma 3.2.1.** For every  $\alpha$  and  $\beta$  of  $B_0$ , (1)  $R(\alpha + \beta) = R(\alpha) + R(\beta) - R(\alpha)R(\beta)$ , (2)  $R(\alpha\beta) = R(\alpha)R(\beta)$ , (3)  $R(\alpha \oplus \beta) = R(\alpha) + R(\beta) - 2R(\alpha)R(\beta)$ , (4)  $R(\alpha') = 1 - R(\alpha)$ .

*Proof.* The lemma is proved by a perfect induction as shown in Table below, where we put  $A = R(\alpha)$  and  $B = R(\beta)$ .

TABLE 3.2.1

$\alpha(A)$	$\beta(B)$	$AB$	$\alpha\beta$	$A+B$	$A+B-AB$	$\alpha+\beta$	$A+B-2AB$	$\alpha\oplus\beta$	$1-A$	$\alpha'$
0	0	0	0	0	0	0	0	0	1	1
1	0	0	0	1	1	1	1	1	0	0
0	1	0	0	1	1	1	1	1		
1	1	1	1	2	1	1	0	0		

Now we define a one-to-one mapping  $\alpha \leftrightarrow \delta$  from  $B_0$  onto the class  $[1, -1]$  by  $\delta = 1 - 2R(\alpha)$ , then we have:

**Theorem 3.2.1.** If  $\alpha \leftrightarrow \delta = 1 - 2R(\alpha)$  and  $\beta \leftrightarrow \varepsilon = 1 - 2R(\beta)$ , then (1)  $\alpha + \beta \leftrightarrow (-1 + \delta + \varepsilon + \delta\varepsilon)/2$ , (2)  $\alpha\beta \leftrightarrow (1 + \delta + \varepsilon - \delta)/2$ , (3)  $\alpha \oplus \beta \leftrightarrow \delta\varepsilon$ , (4)  $\alpha' \leftrightarrow -\delta$ , (5)  $0 \leftrightarrow 1$ ,  $1 \leftrightarrow -1$ .

*Proof.* Proof of (5): Obvious.

*Proof of (4):*  $\alpha' \leftrightarrow 1 - 2R(\alpha')$ . But  $R(\alpha') = 1 - R(\alpha)$  by (4) of Lemma 3.2.1. Hence  $\alpha' \leftrightarrow 1 - 2(1 - R(\alpha)) = 2R(\alpha) - 1 = -\delta$ .

*Proof of (3):*  $\alpha \oplus \beta \leftrightarrow 1 - 2R(\alpha \oplus \beta)$ . But  $R(\alpha \oplus \beta) = R(\alpha) + R(\beta) - 2R(\alpha)R(\beta)$

by (3) Of Lemma 3.2.1. Hence  $\alpha \oplus \beta \leftrightarrow 1 - 2(R(\alpha) + R(\beta) - 2R(\alpha)R(\beta)) = 1 - 2R(\alpha) - 2R(\beta) + 4R(\alpha)R(\beta) = (1 - 2R(\alpha))(1 - 2R(\beta)) = \delta\epsilon$ .

*Proof of (1) and (2):* From (1), (2) and (3) of Lemma 3.2.1, we have

$$R(\alpha + \beta) + R(\alpha\beta) = R(\alpha) + R(\beta) = (1 - \delta)/2 + (1 - \epsilon)/2$$

and  $R(\alpha + \beta) - R(\alpha\beta) = R(\alpha \oplus \beta) = (1 - \delta\epsilon)/2.$

Solving  $R(\alpha + \beta)$  and  $R(\alpha\beta)$  from these equations, we obtain

$$2R(\alpha + \beta) = (3 - \delta - \epsilon - \delta\epsilon)/2$$

and  $2R(\alpha\beta) = (1 - \delta - \epsilon + \delta\epsilon)/2.$

It follows that  $\alpha + \beta \leftrightarrow 1 - 2R(\alpha + \beta) = (-1 + \delta + \epsilon + \delta\epsilon)/2$

and  $\alpha\beta \leftrightarrow 1 - 2R(\alpha\beta) = (1 + \delta + \epsilon - \delta\epsilon)/2.$

Now, in order to obtain a real matrix representation of Boolean functions, let us apply the mapping to every diagonal element of Boolean representation matrices. Then we have the following theorem where  $F^*$  and  $Tr(F)$  is the *transpose* and the *trace* of the matrix  $F$  respectively and  $I$  is the unit matrix.

**Theorem 3.2.2.** Associate with each Boolean function  $f = \sum^{\oplus} \alpha_i a_i$  a  $2^n \times 2^n$  matrix  $F = (\delta_i \delta_{ik})$  where  $\delta_i = 1 - 2R(\alpha_i)$ , then  $B_n$  is represented isomorphically by the class of all  $2^n \times 2^n$  diagonal matrices with diagonal elements 1 or -1, where, if  $f \leftrightarrow F$  and  $g \leftrightarrow G$ ,

- (1)  $f + g \leftrightarrow (-I + F + G + FG)/2,$
- (2)  $fg \leftrightarrow (I + F + G - FG)/2,$
- (3)  $f \oplus g \leftrightarrow FG$
- (4)  $f' \leftrightarrow -F,$
- (5)  $0 \leftrightarrow I, 1 \leftrightarrow -I,$
- (6)  $FG = GF,$
- (7)  $FF^* = I$
- (8)  $Tr(F) = d(f') - d(f) = 2(2^{n-1} - d(f)).$

*Proof.* The proof is easy by Theorem 3.1.1 and Theorem 3.2.1.

The real matrix representation obtained above is, however, a merely special one of such representations. Indeed, there are infinitely many of others, since, transforming every matrix  $F$  into  $\bar{F} = TFT^{-1}$  by any non-singular matrix  $T$ , we can obtain another representation which is equivalent to the original one in the sense that it satisfies the eight properties.

### 3.2. Coordinate Representation of Boolean Functions

In this section, we shall introduce a coordinate representation of Boolean functions. The representation will be derived in a natural manner from a particular real matrix representation as follows.

First, consider all the  $2^n$  odd linear functions  $y_0, y_1, \dots, y_{2^n-1}$ , where  $y_0 = 0$

and  $y_\lambda \oplus y_\mu = y_{\lambda \oplus \mu}$ . Let  $Y_\lambda = (\eta_{\lambda i} \eta_{ik})$  be the real representation matrix of  $y_\lambda$ , and let the matrix  $T = 2^{-n/2}(\eta_{\lambda i})$  be considered.

*Lemma 3.3.1. The matrix  $T$  is orthogonal, i.e.,  $TT^* = 1$ .*

*Proof.* Let us evaluate  $2^n$  times the  $(\lambda, \mu)$  element of  $TT^*$ . It is given by  $\sum_{i=0}^{2^n-1} \eta_{\lambda i} \eta_{\mu i} = \text{Tr}(Y_\lambda Y_\mu)$ . But, by (3) of Theorem 3.2.2,  $Y_\lambda Y_\mu = Y_{\lambda \oplus \mu}$ . Therefore  $\sum \eta_{\lambda i} \eta_{\mu i} = \text{Tr}(Y_{\lambda \oplus \mu}) = 2(2^{n-1} - d(y_{\lambda \oplus \mu}))$  by (8) of Theorem 3.2.2. But  $d(y_\lambda) = 2^{n-1}(1 - \delta_{\lambda 0})$  from Theorem 2.2.1. Hence we have  $\sum \eta_{\lambda i} \eta_{\mu i} = 2^n \delta_{\lambda \mu}$ , because  $\delta_{\lambda \oplus \mu 0} = \delta_{\lambda \mu}$ . This completes the proof.

Now we transform every real representation matrix  $F$  into  $\bar{F} = TFT^*$  by the orthogonal matrix  $T$ , then a new real matrix representation will be obtained. Let us evaluate the matrices  $\bar{F}$ . Putting  $f = \sum_i^{\oplus} \alpha_i a_i$  and  $F = (\delta_i \delta_{ik})$  where  $\delta_i = 1 - 2R(\alpha_i)$  as before,  $2^n$  times the  $(\lambda, \mu)$  element of  $\bar{F}$  is given by:

$$\sum_{i,k} \eta_{\lambda i} \delta_i \delta_{ik} \eta_{\mu k} = \sum_i \delta_i \eta_{\lambda i} \eta_{\mu i} = \text{Tr}(F \oplus Y_{\lambda \oplus \mu}) = 2(2^{n-1} - d(f \oplus y_{\lambda \oplus \mu})).$$

Hence the  $(\lambda, \mu)$  element of  $\bar{F}$  is  $(2^{n-1} - d(f \oplus y_{\lambda \oplus \mu}))/2^{n-1}$ .

Here it is observed that any row (column) of the matrix  $\bar{F}$  is a permutation of any other row (column). Accordingly, in a sense, any one row or column is sufficient for representing Boolean functions. In view of this fact, we define the *coordinates of a Boolean function  $f$*  by  $2^{n-1}$  times the elements of the first row of the matrix  $\bar{F}$ , and denote them by  $f(y_\lambda)$  ( $\lambda = 0, 1, \dots, 2^{n-1}$ ). Thus, the coordinates of  $f$  are given by:

$$f(y_\lambda) = 2^{n-1} - d(f \oplus y_\lambda). \quad (3.3.1)$$

$$\text{Clearly,} \quad f(0) = 2^{n-1} - d(f), \quad (3.3.2)$$

$$\text{while, for } \lambda \neq 0, \quad f(y_\lambda) = d(fy_\lambda) - d(fy'_\lambda), \quad (3.3.2')$$

since  $d(f \oplus y) = d(fy') + d(f'y) = d(fy') + (d(f'y) + d(fy)) - d(fy) = d(fy') - d(fy) + d((f+f')y) = d(fy') - d(fy) + d(f') = d(fy') - d(fy) + 2^{n-1}$  for any odd linear function  $y$  other than 0.

*Theorem 3.3.1. Coordinates of a Boolean function are integers, the sum of whose squares is  $2^{2(n-1)}$ , and, for  $n \geq 2$ , their parity coincides with that of the dimension of the function.*

*Proof. First half:* Obvious from the definition of the coordinates and (7) of Theorem 3.2.2.

*Second half:* For  $n \geq 2$ ,  $f(0) = 2^{n-1} - d(f) \equiv d(f) \pmod{2}$ , while  $f(y_\lambda) = d(fy_\lambda) - d(fy'_\lambda) \equiv d(fy_\lambda) + d(fy'_\lambda) = d(f) \pmod{2}$ .

We shall now present theorems showing how Boolean operations and relations are represented by the coordinates.

*Theorem 3.3.2. For any functions  $f$  and  $g$ ,*

- (1)  $(f+g)(y_\lambda) = (-2^{n-1}\delta_{\lambda 0} + f(y_\lambda) + g(y_\lambda) + (f \oplus g)(y_\lambda))/2$ ,
- (2)  $(fg)(y_\lambda) = (2^{n-1}\delta_{\lambda 0} + f(y_\lambda) + g(y_\lambda) - (f \oplus g)(y_\lambda))/2$ ,
- (3)  $(f \oplus g)(y_\lambda) = \sum_{\mu} f(y_{\lambda \oplus \mu})g(y_\mu)/2^{n-1} = \sum_{\mu} f(y_\mu)g(y_{\lambda \oplus \mu})/2^{n-1}$ ,
- (4)  $f'(y_\lambda) = -f(y_\lambda)$ ,
- (5)  $0(y_\lambda) = 2^{n-1}\delta_{\lambda 0}$ ,  $1(y_\lambda) = -2^{n-1}\delta_{\lambda 0}$ .

*Proof.* Obvious from Theorem 3.2.2 and the definition of the coordinates.

*Theorem 3.3.3. For any functions  $f$  and  $g$ ,*

- (1)  $(f+g)(y_\lambda) = -2^{n-1}\delta_{\lambda 0} + f(y_\lambda) + g(y_\lambda)$

*if and only if  $fg=0$ , while*

- (2)  $(fg')(y_\lambda) = 2^{n-1}\delta_{\lambda 0} + f(y_\lambda) - g(y_\lambda)$

*if and only if  $f \geq g$ .*

*Proof. First half:* If  $fg=0$ , then we have

$$(f+g)(y_\lambda) = (-2^{n-1}\delta_{\lambda 0} + f(y_\lambda) + g(y_\lambda) + (f \oplus g)(y_\lambda))/2$$

and  $(fg)(y_\lambda) = (2^{n-1}\delta_{\lambda 0} + f(y_\lambda) + g(y_\lambda) - (f \oplus g)(y_\lambda))/2 = 2^{n-1}\delta_{\lambda 0}$

from (1), (2) and (5) of Theorem 3.3.2. Eliminating  $(f \oplus g)(y_\lambda)$  from the above equations, the desired relation (1) will be obtained. Conversely, if we assume (1), (1) of Theorem 3.3.2 yields  $(f+g)(y_\lambda) = (f \oplus g)(y_\lambda)$ , i.e.,  $f+g=f \oplus g$ . But  $f+g=f \oplus g \oplus fg$ . Hence  $fg=0$ .

*Second half:*  $f \geq g$  is equivalent to  $f'g=0$ , so we have

$$(f'+g)(y_\lambda) = -2^{n-1}\delta_{\lambda 0} + f'(y_\lambda) + g(y_\lambda)$$

from the first half. But  $(f'+g)'=fg'$ . Therefore, by (4) or Theorem 3.3.2, we obtain (2).

Here let us pause for a while and make some conventions for the notation of linear functions. In the first place, we shall use the letters  $y$  and  $z$  with or without suffixes exclusively for odd linear functions hereafter. In the second place, we shall denote linear functions by symbols such as  $y^\delta$  where  $\delta = \pm 1$  and  $y^\delta$  is  $y$  or  $y'$  according as  $\delta$  is 1 or  $-1$ . The Greek letters  $\delta$  and  $\varepsilon$  will be reserved for this purpose.

Now, for the sake of illustration, the coordinates of linear functions and atoms will be examined. First, for any linear function  $y_\mu^\varepsilon$ , we have

$$y_\mu^\varepsilon(y_\lambda) = 2^{n-1}\varepsilon\delta_{\mu\lambda} \quad (3.3.3)$$

from (3.3.1) and (4) of Theorem 3.3.2. Next, let  $a$  be an atom, then, since  $d(a)=1$  and either  $d(fa)=1$  and  $d(f'a)=0$  or  $d(fa)=0$  and  $d(f'a)=1$  for every  $f$ , we have

$$a(0) = 2^{n-1} - 1, |a(y_\lambda)| = 1 \quad (\lambda \neq 0) \quad (3.3.4)$$

from (3.3.2) and (3.3.2'). Furthermore, if we put  $a = x_1^{\varepsilon_1} x_2^{\varepsilon_2} \cdots x_n^{\varepsilon_n}$ , we obtain

$$a(x_a \oplus x_b \oplus \cdots \oplus x_s) = (-1)^{k+1 \varepsilon_a \varepsilon_b \cdots \varepsilon_s}, \quad (3.3.5)$$

where the length of  $x_a \oplus x_b \oplus \cdots \oplus x_s$  is assumed to be  $k$ . (3.3.5) will be proved by a mathematical induction on the values of  $k$  by the help of the property that, if  $a \leq y^{\delta}$  and  $a \leq z^{\delta}$ , then  $a \leq (y \oplus z)^{-\delta \varepsilon}$ . Note that (3.3.4) and (3.3.5) yields, for the atom  $a_0 = x'_1 x'_2 \cdots x'_n$ ,

$$a_0(y_\lambda) = 2^{n-1} \delta_{\lambda 0} - 1. \quad (3.3.6)$$

In Table 3.3.1 and Table 3.3.2 below, the coordinates of all atoms of three and four variables are shown. The coordinates are grouped into sets in accordance with the lengths of odd linear functions, and, in each set, are arranged in the lexicographical order. Thus, for example, the column  $\frac{1}{2}$  contains the coordinates for  $x_i \oplus x_2$ . Negative signs are denoted by bars put over the digits. These conventions will be observed from now on.

TABLE 3.3.1

	0	1	2	3	1	1	2	2
					2	3	3	3
0	3	1	1	1	1	1	1	1
1	3	1	1	1	1	1	1	1
2	3	1	1	1	1	1	1	1
3	3	1	1	1	1	1	1	1
4	3	1	1	1	1	1	1	1
5	3	1	1	1	1	1	1	1
6	3	1	1	1	1	1	1	1
7	3	1	1	1	1	1	1	1

TABLE 3.3.2

	0	1	2	3	4	1	1	1	2	2
						2	2	3	3	3
0	7	1	1	1	1	1	1	1	1	1
1	7	1	1	1	1	1	1	1	1	1
2	7	1	1	1	1	1	1	1	1	1
3	7	1	1	1	1	1	1	1	1	1
4	7	1	1	1	1	1	1	1	1	1
5	7	1	1	1	1	1	1	1	1	1
6	7	1	1	1	1	1	1	1	1	1
7	7	1	1	1	1	1	1	1	1	1
8	7	1	1	1	1	1	1	1	1	1
9	7	1	1	1	1	1	1	1	1	1
10	7	1	1	1	1	1	1	1	1	1
11	7	1	1	1	1	1	1	1	1	1
12	7	1	1	1	1	1	1	1	1	1
13	7	1	1	1	1	1	1	1	1	1
14	7	1	1	1	1	1	1	1	1	1
15	7	1	1	1	1	1	1	1	1	1

Now we shall show how to calculate coordinates of an arbitrary function  $f$  from those of atoms. To begin with, we rewrite (1) of Theorem 3.3.3 as:

$$(f + g)(y_\lambda) - 2^{n-1} \delta_{\lambda 0} = (f(y_\lambda) - 2^{n-1} \delta_{\lambda 0}) + (g(y_\lambda) - 2^{n-1} \delta_{\lambda 0}),$$

then it can be proved by induction that, if  $f_i f_j = 0$  for every  $i \neq j$ ,

$$\left( \sum_i f_i \right) (y_\lambda) - 2^{n-1} \delta_{\lambda 0} = \sum_i (f_i(y_\lambda) - 2^{n-1} \delta_{\lambda 0}),$$

[illegible]



may only reverse their signs. The final result is shown in the bottom line of Table 3.3.4.

### 3.4. Some Useful Theorems

In this section, some useful theorems for the study of Boolean functions by means of the coordinate representation will be derived.

*Theorem 3.4.1. For any function  $f$  and any linear function  $y_\mu^{\varepsilon_\mu}$ , a necessary and sufficient condition that  $f \leq y_\mu^{\varepsilon_\mu}$  is*

$$2^{n-1}\delta_{\lambda 0} - f(y_\lambda) = -\varepsilon_\mu(2^{n-1}\delta_{\lambda\mu} - f(y_\lambda \oplus y_\mu)) \quad (3.4.1)$$

for every  $\lambda$ . The set of all odd linear functions  $y_\mu$  such that  $f \leq y_\mu^{\varepsilon_\mu}$  forms a subgroup  $L(f)$  of  $L_n$ . Let its base be  $[y_1, y_2, \dots, y_k]$ , then we have

$$f(y_\alpha \oplus y_\beta \oplus \dots \oplus y_\sigma) = -(-\varepsilon_\alpha)(-\varepsilon_\beta) \dots (-\varepsilon_\sigma)d(f). \quad (3.4.2)$$

*Proof.* From (2) of Theorem 3.3.2 and (3.3.3), we have

$$(fy_\mu^{\varepsilon_\mu})(y_\lambda) = (2^{n-1}\delta_{\lambda 0} + f(y_\lambda) + 2^{n-1}\varepsilon_\mu\delta_{\lambda\mu} - \varepsilon_\mu f(y_\lambda \oplus y_\mu))/2.$$

Therefore (3.4.1) is equivalent to  $(fy_\mu^{\varepsilon_\mu})(y_\lambda) = f(y_\lambda)$ , i.e., to  $f \leq y_\mu^{\varepsilon_\mu}$ . Strictly speaking, only

$$f(y_\mu) = \varepsilon_\mu d(f) \quad (3.4.3)$$

is sufficient for  $f \leq y_\mu^{\varepsilon_\mu}$ , since it implies  $(fy_\mu^{\varepsilon_\mu})(0) = f(0)$ , i.e.,  $d(fy_\mu^{\varepsilon_\mu}) = d(f)$ . Thus the first half is established.

Next, let  $L(f)$  be the set all odd linear functions  $y_\mu$  such that  $f \leq y_\mu^{\varepsilon_\mu}$ . Clearly  $0 \in L(f)$  because  $f \leq 1 = 0'$ . Further, since  $f \leq y_\alpha^{\varepsilon_\alpha}$  and  $f \leq y_\beta^{\varepsilon_\beta}$  imply  $f \leq (y_\alpha \oplus y_\beta)^{-\varepsilon_\alpha \varepsilon_\beta}$ , it follows that, if  $y_\alpha \in L(f)$  and  $y_\beta \in L(f)$ ,  $y_\alpha \oplus y_\beta \in L(f)$ . Hence  $L(f)$  is a subgroup of  $L_n$ . When we use the fact that  $f(y_\alpha \oplus y_\beta) = -\varepsilon_\alpha \varepsilon_\beta d(f)$  if  $f(y_\alpha) = \varepsilon_\alpha d(f)$  and  $f(y_\beta) = \varepsilon_\beta d(f)$ , (3.4.2) can be proved by the induction on the length of linear function. This completes the proof.

The theorem indicates that, to each function  $f$ , there corresponds a certain group  $L(f)$  of odd linear functions. For example, consider the four-variable function  $f$  with coordinates 4 4000 0004 00 4000 0. Since the first coordinate means that  $d(f) = 8 - 4 = 4$ , it follows that  $f \leq x_1$ ,  $f \leq (x_2 \oplus x_3)'$  and  $f \leq x_1 \oplus x_2 \oplus x_3$ . Therefore,  $L(f)$  is given by the four-element group  $[0, x_1, x_2 \oplus x_3, x_1 \oplus x_2 \oplus x_3]$ .

Now we shall calculate the coordinates of functions of the form  $f = y_1^{\varepsilon_1} y_2^{\varepsilon_2} \dots y_k^{\varepsilon_k}$  for later uses, where  $[y_1, y_2, \dots, y_k]$  is any set of  $k$  independent odd linear functions. The result is as follows.

*Theorem 3.4.2. Let  $[y_1, y_2, \dots, y_k]$  be any set of mutually independent odd linear functions, then, putting  $f = y_1^{\varepsilon_1} y_2^{\varepsilon_2} \dots y_k^{\varepsilon_k}$ , we have*

$$f(0) = 2^{n-1} - 2^{n-k}, \quad (3.4.4)$$

$$f(y_\lambda) = -(-\varepsilon_\alpha)(-\varepsilon_\beta) \dots (-\varepsilon_\sigma) 2^{n-k} \quad (3.4.5)$$

for  $y_\lambda = y_\alpha \oplus y_\beta \oplus \cdots \oplus y_\sigma \in L(f)$ , and

$$f(y_\lambda) = 0 \quad (3.4.6)$$

for every  $y_\lambda \notin L(f)$ , where  $L(f)$  is the group determined by the base  $[y_1, y_2, \dots, y_k]$ .

*Proof.* Suppose that (3.4.4) is proved, then, since  $d(f) = 2^{n-k}$ , (3.4.5) is evident from (3.4.2). Again suppose that (3.4.4) and (3.4.5) are proved, then, from Theorem 3.3.1, we have

$$\begin{aligned} 2^{2(n-1)} &= \sum_{L_n} f^2(y_\lambda) = \sum_{L(f)} f^2(y_\lambda) + \sum_{L(f)^o} f^2(y_\lambda) = (2^{n-1} - 2^{n-k})^2 + (2^k - 1)2^{2(n-k)} \\ &\quad + \sum_{L(f)^o} f^2(y_\lambda) = 2^{2(n-1)} + \sum_{L(f)^o} f^2(y_\lambda) \end{aligned}$$

Hence  $\sum_{L(f)^o} f^2(y_\lambda) = 0$ , i.e.,  $f(y_\lambda) = 0$  for every  $y_\lambda \notin L(f)$ . Therefore it suffices to prove (3.4.4).

First, (3.4.4) is seen to be valid for  $k=1$  by (3.3.3). Next, assume that it is valid for a certain value of  $k$ . Then putting  $f = y_1^{\varepsilon_1} y_2^{\varepsilon_2} \cdots y_k^{\varepsilon_k}$  and  $g = y_{k+1}^{\varepsilon_{k+1}}$ , we obtain

$$(fg)(0) = (2^{n-1} + f(0) + g(0) - (f \oplus g)(0))/2$$

from (2) of Theorem 3.3.2. But  $f(0) = 2^{n-1} - 2^{n-k}$  and  $g(0) = 0$  by (3.3.3). Further, by (3) of Theorem 3.3.2 and (3.3.3), we obtain

$$(f \oplus g)(0) = \varepsilon_{k+1} f(y_{k+1}),$$

which is 0 by (3.4.6). Therefore  $(fg)(0)$  is given by

$$(fg)(0) = (2^{n-1} + 2^{n-1} - 2^{n-k})/2 = 2^{n-1} - 2^{n-k-1}.$$

Thus, by induction, (3.4.4) is proved.

The theorem implies that, for any set of  $n$  mutually independent odd linear functions  $[y_1, y_2, \dots, y_n]$ , every function of the form  $y_1^{\varepsilon_1} y_2^{\varepsilon_2} \cdots y_n^{\varepsilon_n}$  has the dimension 1, or in other words, every function of the form  $y_1^{\varepsilon_1} y_2^{\varepsilon_2} \cdots y_n^{\varepsilon_n}$  is an atom. Moreover, since there are  $2^n$  distinct functions of this form, it is concluded that every atom can be expressed in this form. This remarkable property of linear functions will be used to introduce a new transformation of Boolean functions in Section 3.8.

*Theorem 3.4.3. Let  $L$  be a  $k$ -dimensional subgroup of  $L_n$ , then, for any function  $f$  and any odd linear function  $y_\lambda$ , we have*

$$\sum_{y_\mu \in L} \delta_\mu f(y_\lambda \oplus y_\mu) \equiv 2^{n-1} \pmod{2^k} \quad (3.4.7)$$

$$\text{and} \quad \left| \sum_{y_\mu \in L} \delta_\mu f(y_\lambda \oplus y_\mu) \right| \leq 2^{n-1}, \quad (3.4.8)$$

where  $\delta_\mu = 1$  or  $-1$  and  $\delta_{\mu \oplus \nu} = \delta_\mu \delta_\nu$  for any  $y_\mu$  and  $y_\nu$  in  $L$ .

*Proof.* Let  $[y_1, y_2, \dots, y_k]$  be any base of  $L$ . Then putting  $g = y_1^{\varepsilon_1} y_2^{\varepsilon_2} \cdots y_k^{\varepsilon_k}$  in (3) of Theorem 3.3.2 and using (3.4.6), we have

$$(f \oplus g)(y_\lambda) = \sum_{y_\mu \in L} g(y_\mu) f(y_\lambda \oplus y_\mu) / 2^{n-1}.$$

Now, from (3.4.4) and (3.4.5), it will be observed that  $g(y_\mu)$  for  $y_\mu \in L$  can be given by

$$g(y_\mu) = 2^{n-1} \delta_{\mu 0} - \delta_\mu 2^{n-k},$$

where  $\delta_0 = 1$ ,  $\delta_i = -\varepsilon_i$  ( $i = 1, 2, \dots, k$ ) and  $\delta_{\mu \oplus \nu} = \delta_\mu \delta_\nu$  for any  $y_\mu$  and  $y_\nu$  in  $L$ . Hence we obtain

$$(f \oplus g)(y_\lambda) = f(y_\lambda) - \sum_{y_\mu \in L} \delta_\mu f(y_\lambda \oplus y_\mu) / 2^{k-1}.$$

This expression for  $(f \oplus g)(y_\lambda)$ , when inserted into (2) of Theorem 3.3.2, yields

$$(fg)(y_\lambda) = (2^{n-1} \delta_{\lambda 0} + g(y_\lambda) + \sum_{y_\mu \in L} \delta_\mu f(y_\lambda \oplus y_\mu) / 2^{k-1}) / 2.$$

First, let us assume that  $y_\lambda \in L$ , then, since  $g(y_\lambda) = 2^{n-1} \delta_{\lambda 0} - \delta_\lambda 2^{n-k}$ , the above expression will be rewritten as

$$\sum_{y_\mu \in L} \delta_\mu f(y_\lambda \oplus y_\mu) = 2^{n-1} \delta_\lambda - 2^k (2^{n-1} \delta_{\lambda 0} - (fg)(y_\lambda)).$$

But  $fg \leq y_\lambda^{-\delta_\lambda}$  because  $g = y_1^{\varepsilon_1} y_2^{\varepsilon_2} \cdots y_k^{\varepsilon_k} \leq y_\lambda^{-\delta_\lambda}$ . Hence it follows that

$$2^{n-1} \delta_{\lambda 0} - (fg)(y_\lambda) = \delta_\lambda d(fg)$$

from (3.4.1). Therefore we obtain

$$\sum_{y_\mu \in L} \delta_\mu f(y_\lambda \oplus y_\mu) = \delta_\lambda (2^{n-1} - 2^k d(fg)). \quad (3.4.9)$$

Since  $0 \leq d(fg) \leq 2^{n-k}$  by (3.4.4), (3.4.7) and (3.4.8) are verified.

Next, assume that  $y_\lambda \notin L$ , then we have

$$(fg)(y_\lambda) = \sum_{y_\mu \in L} \delta_\mu f(y_\lambda \oplus y_\mu) / 2^k,$$

and consequently

$$\sum_{y_\mu \in L} \delta_\mu f(y_\lambda \oplus y_\mu) = 2^k (d(fgy_\lambda) - d(fgy'_\lambda)). \quad (3.4.10)$$

But  $k \leq n-1$  because there exists  $y_\lambda$  such that  $y_\lambda \notin L$ . Hence we have

$$\sum_{y_\mu \in L} \delta_\mu f(y_\lambda \oplus y_\mu) \equiv 0 \equiv 2^{n-1} \pmod{2^k}.$$

Further,  $-2^{n-k-1} \leq d(fgy_\lambda) - d(fgy'_\lambda) \leq 2^{n-k-1}$ , since  $0 \leq d(fgy_\lambda) \leq 2^{n-k-1}$  and  $0 \leq d(fgy'_\lambda) \leq 2^{n-k-1}$  by (3.4.4). Therefore (3.4.8) is proved.

*Theorem 3.4.4.* For any function, the sum of the absolute values of any two coordinates never exceeds  $2^{n-1}$ .

*Proof.* Let  $y_\lambda$  and  $y_\mu$  be any two odd linear functions, then, putting  $L = [0, y_\lambda \oplus y_\mu]$  and  $\delta_{\lambda \oplus \mu} = \text{Sign}(f(y_\lambda)f(y_\mu))$  in (3.4.8), we obtain

$$|f(y_\lambda) + \delta_{\lambda \oplus \mu} f(y_\mu)| = |f(y_\lambda)| + |f(y_\mu)| \leq 2^{n-1}.$$

*Theorem 3.4.5.* For every function  $f$  of three or more than three variables, either  $f(y_\lambda) \equiv f(0) \pmod{4}$  for every  $\lambda$ , or else the number of coordinates such that  $f(y_\lambda) \equiv f(0) \pmod{4}$  and the number of coordinates such that  $f(y_\lambda) \equiv f(0) + 2 \pmod{4}$  are equal to one another.

*Proof.* Putting  $L = [0, y_\mu, y_\nu, y_\mu \oplus y_\nu]$ ,  $y_\lambda = 0$  and  $\delta_\mu = \delta_\nu = 1$  in (3.4.7), we obtain

$$f(0) + f(y_\mu) + f(y_\nu) + f(y_\mu \oplus y_\nu) \equiv 0 \pmod{4} \quad (3.4.11)$$

because  $2^{n-1} \equiv 0 \pmod{4}$  for  $n \geq 3$ . This can be satisfied if  $f(y_\lambda) \equiv f(0) \pmod{4}$  for every  $\lambda$ . In fact, such functions exist. An example is the function 0 whose first coordinates is  $2^{n-1}$  and all others are 0. On the other hand, if there exists a certain  $y_\mu$  such that  $f(y_\mu) \equiv f(0) + 2 \pmod{4}$ , then (3.4.11) can be rewritten as

$$f(y_\nu) + f(y_\mu \oplus y_\nu) \equiv 2f(0) + 2 \pmod{4}.$$

Hence we have  $f(y_\mu \oplus y_\nu) \equiv f(0) + 2 \pmod{4}$  or  $f(y_\mu \oplus y_\nu) \equiv f(0) \pmod{4}$  according to whether  $f(y_\nu) \equiv f(0) \pmod{4}$  or  $f(y_\nu) \equiv f(0) + 2 \pmod{4}$ . Since  $L_n$  is partitioned into  $2^{n-1}$  rest classes  $[y_\nu, y_\mu \oplus y_\nu]$  by its subgroup  $[0, y_\mu]$ , the latter half of the theorem is proved. An example of functions for which the latter half holds is given by the atom  $x_1 x_2 x_3$  whose coordinates are 3 1 1 1  $\bar{1} \bar{1} \bar{1}$  1.

We now proceed to the essential problem of characterizing the coordinates of Boolean functions.

*Theorem 3.4.6.* A necessary and sufficient condition that an ordered  $2^n$ -tuple  $[f(y_\lambda)]$ ,  $(y_\lambda \in L_n)$  be coordinates of a Boolean function is

$$\sum_{L_n} \delta_\lambda f(y_\lambda) = \pm 2^{n-1} \quad (3.4.12)$$

for every possible combination of values of  $\delta_\lambda = \pm 1$  restricted by  $\delta_{\mu \oplus \nu} = \delta_\mu \delta_\nu$  for any  $y_\mu$  and  $y_\nu$  in  $L_n$ .

*Proof.* *Necessity:* Obvious because (3.4.7) and (3.4.8) are reduced to (3.4.12) when  $k = n$  and  $y_\lambda = 0$ .

*Sufficiency:* Evidently, a sufficient condition is that the matrix  $F = T^* \bar{F} T$  is diagonal and orthogonal, where  $\bar{F} = (f(y_{\lambda \oplus \mu})/2^{n-1})$  and  $T = 2^{-n/2}(\eta_{\lambda i})$ . Rewriting the condition, we have

$$\pm 2^{n-1} = 2^{-n} \sum_{\lambda, \mu} \eta_{\lambda i} f(y_{\lambda \oplus \mu}) \eta_{\mu i} = 2^{-n} \sum_{\lambda, \mu} \eta_{\lambda i} \eta_{\mu i} f(y_{\lambda \oplus \mu})$$

for every  $i$  ( $i = 0, 1, \dots, 2^n - 1$ ). But  $\eta_{\lambda i} = \pm 1$  and  $\eta_{\lambda i} \eta_{\mu i} = \eta_{\lambda \oplus \mu i}$ . Thus, putting  $\eta_{\lambda i} = \delta_i$ , we obtain

$$\pm 2^{n-1} = 2^{-n} \sum_{\mu} \sum_{\lambda} \delta_{\lambda \oplus \mu} f(y_{\lambda \oplus \mu}) = 2^{-n} 2^n \sum_{\lambda} \delta_{\lambda} f(y_{\lambda}) = \sum_{\lambda} \delta_{\lambda} f(y_{\lambda}).$$

Here it is noted that (3.4.12) provides a way for identifying the Boolean function  $f$  represented by  $[f(y_{\lambda})]$ , since, by (3.4.9), the atom  $y_1^{\varepsilon_1} y_2^{\varepsilon_2} \cdots y_n^{\varepsilon_n}$  is included in  $f$  if and only if (3.4.12) gives the value  $-2^{n-1}$  for  $\delta_i = -\varepsilon_i$  ( $i=1, 2, \dots, n$ ), where  $[y_1, y_2, \dots, y_n]$  is any set of  $n$  mutually independent odd linear functions.

The theorem is stated with a theoretical simplicity, but it is cumbersome to use, because, in general,  $2^n$  trials will be needed to determine whether a given  $2^n$ -tuple represents a Boolean function or not. Accordingly, some deformations will be desirable to reduce the number of trials. One of the solutions to the above problem is given by the following theorem which reduces the number of trials down to  $2^{n-1}$ .

*Theorem 3.4.7. A necessary and sufficient condition that an ordered  $2^n$ -tuple  $[f(y_{\lambda})]$  ( $y_{\lambda} \in L_n$ ) be coordinates of a Boolean function is*

$$\sum_{L^*} \delta_{\lambda} f(y_{\lambda}) = \pm 2^{n-1} \quad \text{and} \quad \sum_{L^*} \delta_{\lambda} f(y_{\lambda} \oplus y_n) = 0, \quad (3.4.13)$$

$$\text{or} \quad \sum_{L^*} \delta_{\lambda} f(y_{\lambda}) = 0 \quad \text{and} \quad \sum_{L^*} \delta_{\lambda} f(y_{\lambda} \oplus y_n) = \pm 2^{n-1} \quad (3.4.13')$$

for every combination of values of  $\delta_{\lambda} = \pm 1$  such that  $\delta_{\mu \oplus \nu} = \delta_{\mu} \delta_{\nu}$  for any  $y_{\mu}$  and  $y_{\nu}$  in  $L^*$ , where  $[y_1, y_2, \dots, y_n]$  is a base of  $L_n$  and  $L^*$  is the subgroup of  $L_n$  determined by the base  $[y_1, y_2, \dots, y_{n-1}]$ .

*Proof.*  $L_n$  is partitioned into two rest classes  $L^*$  and  $L^{**} = y_n \oplus L^*$ , so the condition (3.4.12) is equivalent to

$$\sum_{L_n} \delta_{\lambda} f(y_{\lambda}) = \sum_{L^*} \delta_{\lambda} f(y_{\lambda}) + \sum_{L^{**}} \delta_{\lambda} f(y_{\lambda}) = \sum_{L^*} \delta_{\lambda} f(y_{\lambda}) + \delta_n \sum_{L^*} \delta_{\lambda} f(y_{\lambda} \oplus y_n) = \pm 2^{n-1},$$

whence the sufficiency is evident. The necessity is also proved by Theorem 3.4.3, because it tells us that

$$\sum_{L^*} \delta_{\lambda} f(y_{\lambda}) = \pm 2^{n-1} \text{ or } 0$$

$$\text{and} \quad \sum_{L^*} \delta_{\lambda} f(y_{\lambda} \oplus y_n) = \pm 2^{n-1} \text{ or } 0.$$

Note here that (3.4.13) and (3.4.13') provide a way for identifying the Boolean function  $f$  represented by  $[f(y_{\lambda})]$ , since, by (3.4.9) and (3.4.10), the atom  $y_1^{\varepsilon_1} y_2^{\varepsilon_2} \cdots y_n^{\varepsilon_n}$  is included in  $f$  if and only if either

$$\sum_{L^*} \delta_{\lambda} f(y_{\lambda}) = -2^{n-1}$$

$$\text{or} \quad \sum_{L^*} \delta_{\lambda} f(y_{\lambda}) = 0 \quad \text{and} \quad \sum_{L^*} \delta_{\lambda} f(y_{\lambda} \oplus y_n) = 2^{n-1} \varepsilon_n$$

for  $\delta_i = -\varepsilon_i$  ( $i=1, 2, \dots, n-1$ ).

As another interesting deformation of Theorem 3.4.6, we have:

*Theorem 3.4.8. A necessary and sufficient condition that an ordered  $2^n$ -tuple  $[f(y_\lambda)]$  ( $y_\lambda \in L_n$ ) be coordinates of a Boolean function is that the matrix  $\overline{F} = (f(y_{\lambda \oplus \mu})/2^{n-1})$  is orthogonal.*

*Proof. Necessity:* Obvious.

*Sufficiency:*

$$(\sum_{\lambda \in L_n} \delta_\lambda f(y_\lambda))^2 = \sum_{\lambda} \sum_{\mu} \delta_\lambda \delta_\mu f(y_\lambda) f(y_\mu).$$

Putting  $y_\lambda \oplus y_\mu = y_\nu$ , we obtain

$$(\sum_{\lambda} \delta_\lambda f(y_\lambda))^2 = \sum_{\lambda} \sum_{\nu} \delta_\nu f(y_\lambda) f(y_\lambda \oplus y_\nu) = \sum_{\lambda} f(y_\lambda)^2 + \sum_{\nu \neq 0} \delta_\nu \sum_{\lambda} f(y_\lambda) f(y_\lambda \oplus y_\nu).$$

But  $\overline{F}$  is orthogonal. Hence we have

$$\sum_{\lambda} f(y_\lambda)^2 = 2^{2(n-1)}$$

and

$$\sum_{\lambda} f(y_\lambda) f(y_\lambda \oplus y_\nu) = 0.$$

Therefore

$$\sum \delta_\lambda f(y_\lambda) = \pm 2^{n-1}.$$

This completes the proof.

### 3.5. Coordinate Representation of Symmetry Structures

In Section 2.3, we have referred to the symmetries (symmetric automorphisms) and the symmetry structures of Boolean functions. In this section, we shall return to the same subject and see how the symmetries are expressed and how the symmetry structures can be recognized in the coordinate representation. The following theorem is fundamental for these purposes.

*Theorem 3.5.1. Let  $\sigma$  be a symmetry, then, for any Boolean function  $f$ , we have*

$$(\sigma f)(y_\lambda) = f(\sigma^{-1}y_\lambda). \quad (3.5.1)$$

*Proof.* From (3.3.1), we have

$$f(y_\mu) = 2^{n-1} - d(f \oplus y_\mu).$$

$$\begin{aligned} \text{Accordingly, } (\sigma f)(\sigma y_\mu) &= 2^{n-1} - d(\sigma f \oplus \sigma y_\mu) = 2^{n-1} - d(\sigma(f \oplus y_\mu)) \\ &= 2^{n-1} - d(f \oplus y_\mu) = f(y_\mu). \end{aligned}$$

Hence, putting  $\sigma y_\mu = y_\lambda$ , we obtain (3.5.1).

The theorem expresses in a simple form the effects of a symmetry on the coordinates of a Boolean function. If a permutation operator is denoted by  $\pi$ , a reflexion operator by  $\gamma$ , and an odd linear function by  $y_\lambda$ , then  $\pi y_\lambda$  is an odd linear function of the same length as  $y_\lambda$  and  $\gamma y_\lambda$  is  $y_\lambda$  or  $y'_\lambda$ . Therefore the effects

pointed out in Theorem 3.5.1 are a permutation and/or changes of signs of the coordinates. Precisely speaking, the effect of a permutation operator is a permutation of the coordinates within each set corresponding to the set of odd linear functions having the same length and that of a reflexion operator is changes of signs of the coordinates.

*Example 3.5.1.* Apply the symmetry  $\sigma = [23](13)$  to the three-variable function  $f = \sum(5, 6, 7)$  whose coordinates are  $1\ 3\ 1\ 1\ \bar{1}\ \bar{1}\ \bar{1}$ .

*Solution.* First, apply  $(13)$  to  $f$ , then, since  $(13)^{-1} = (13)$ , we obtain  $(13)f: 1\ 1\ 1\ 3\ 1\ \bar{1}\ \bar{1}$  by interchanging  $f(x_1) = 3$  with  $f(x_3) = 1$ ,  $f(x_1 \oplus x_2) = \bar{1}$  with  $f(x_2 \oplus x_3) = 1$  and keeping other coordinates unchanged. Next, apply  $[23]$  to  $g = (13)f$ , then, since  $[23]^{-1} = [23]$ , we obtain  $[23]g: 1\ 1\ \bar{1}\ \bar{3}\ \bar{1}\ \bar{1}\ \bar{1}$  by changing the signs of  $g(x_2)$ ,  $g(x_3)$ ,  $g(x_1 \oplus x_2)$  and  $g(x_1 \oplus x_3)$  and keeping others unchanged.

Evidently two functions  $f$  and  $g$  are of the same type (genus) if and only if there exists a symmetry  $\sigma$  such that  $f(\sigma^{-1}y_\lambda) = g(y_\lambda)$  ( $\varepsilon f(\sigma^{-1}y_\lambda) = g(y_\lambda)$ ). But the above condition may be replaced by a somewhat weaker one for functions of odd dimensions.

TABLE 3.5.1

	0	1	2	3	1 1 2	2
					2 3 3	3
$f$	1	3	1	1	$\bar{1}$	$\bar{1}$
$(13)f$	1	1	1	3	$\bar{1}$	$\bar{1}$
$[23](13)f$	1	1	$\bar{1}$	$\bar{3}$	$\bar{1}$	$\bar{1}$

*Theorem 3.5.2.* If  $f$  and  $g$  are functions of an odd dimension, then they are of the same genus if and only if there exists a permutation operator  $\pi$  such that  $|f(\pi^{-1}y_\lambda)| = |g(y_\lambda)|$ .

*Proof.* As will be easily seen, it suffices to prove that, two functions  $f$  and  $g$  of an odd dimension such that  $|f(y_\lambda)| = |g(y_\lambda)|$  are of the same genus.

First, assume  $n \leq 2$ , then every function of an odd dimension is either an atom or an anti-atom. Therefore the theorem is valid because atoms and anti-atoms are of the same genus.

Next, assume  $n \geq 3$ . To begin with, we shall show that any function of an odd dimension can be transformed into a certain standard form by a suitable reflexion operator. Let  $f$  be transformed into  $f^* = \tau f$  by a reflexion operator  $\tau$ , where  $f^*(x_i) \equiv f^*(0) = f(0) \pmod{4}$  ( $i = 1, 2, \dots, n$ ). This is always feasible, because we may prime the variables  $x_i$  such that  $f(x_i) \equiv f(0) + 2 \pmod{4}$ . Now, we see from Theorem 3.4.3.

$$f^*(0) + f^*(x_i) + f^*(x_j) + f^*(x_i \oplus x_j) \equiv 0 \pmod{4},$$

which means  $f^*(y_\lambda) \equiv f^*(0) \pmod{4}$  for every  $y_\lambda$  of the length 2. Again we see from the same theorem

$$f^*(0) + f^*(x_k) + f^*(x_i \oplus x_j) + f^*(x_i \oplus x_j \oplus x_k) \equiv 0 \pmod{4},$$

which means  $f^*(y_\lambda) \equiv f^*(0) \pmod{4}$  for every  $y_\lambda$  of the length 3. Going on in this way, we arrive at the conclusion that

$$f^*(y_\lambda) \equiv f^*(0) \pmod{4}$$

for every  $y_\lambda$ . This function  $f^*$  is called the *standard form* of  $f$ .

Now let  $g^*$  be the standard form of  $g$ , then obviously  $f^*(y_\lambda) = g^*(y_\lambda)$  or  $f^*(y_\lambda) = -g^*(y_\lambda)$  according as  $f(0) = g(0)$  or  $f(0) = -g(0)$ . This completes the proof.

In connection with this theorem, there arises a natural question: "Is the similar proposition true as well for functions of even dimensions?" Unfortunately, however, the answer is "No", because a counter example is found for  $n=4$ . In fact, the four functions shown in Table 3.5.2 below provide a counter example. As is seen, every coordinate of these functions has the absolute value 2, but they are of different types.

TABLE 3.5.2

$\Sigma(0, 7, 11, 13, 14, 15)$ :	2	2	2	2	2	$\bar{2}$	$\bar{2}$	$\bar{2}$	$\bar{2}$	$\bar{2}$	$\bar{2}$	2
$\Sigma(1, 6, 11, 13, 14, 15)$ :	2	2	2	2	2	$\bar{2}$	$\bar{2}$	$\bar{2}$	$\bar{2}$	2	2	2
$\Sigma(3, 5, 10, 13, 14, 15)$ :	2	2	2	2	2	$\bar{2}$	$\bar{2}$	2	2	$\bar{2}$	$\bar{2}$	$\bar{2}$
$\Sigma(3, 7, 11, 12, 13, 14)$ :	2	2	2	2	2	$\bar{2}$	2	2	$\bar{2}$	$\bar{2}$	$\bar{2}$	2

We now consider the problem of analysing the symmetry structures of Boolean functions by their coordinates. To begin with, the problem of recognizing some of the typical symmetry structures will be considered. The following four theorems serve for this purpose.

*Theorem 3.5.3. A necessary condition for a function to be symmetrizable is that all the coordinates appertaining to each set of odd linear functions of the same length have the same absolute value; A necessary and sufficient condition for a function to be symmetric is that all the coordinates appertaining to each set of odd linear functions of the same length have the same value.*

*Proof.* Obvious.

*Theorem 3.5.4. A necessary and sufficient condition for a function of an odd dimension to be symmetrizable is that all the coordinates appertaining to each set of odd linear functions of the same length have the same absolute value.*

*Proof.* Evidently the above condition is necessary. Its sufficiency is proved as follows. First, assume  $n \leq 2$ , then every function of an odd dimension is symmetrizable, because it is an atom or an anti-atom.

Next, assume  $n \geq 3$ , then the standard form of a function satisfying the condition of the theorem is symmetric. Therefore it is symmetrizable.

For example, the three-variable function 2 000 222 0 is symmetric and the four-variable function 3 11 $\bar{1}$  $\bar{1}$   $\bar{1}$ 111 $\bar{1}$  33 $\bar{3}$  $\bar{3}$  3 is symmetrizable. Notice that Table 3.5.2 provides a counter example against the validity of the generalization of Theorem 3.5.4 to even dimensions, because only the first function of the table is symmetric and all the others are not.

*Theorem 3.5.5. A function is independent of a variable if and only if its coordinates corresponding to odd linear functions containing the variable are all 0.*



*Proof.* Obvious.

Thus, for example, the three-variable function  $f: 2\ 220\ \bar{2}00\ 0$  is independent of  $x_3$ , because its coordinates  $f(x_3)$ ,  $f(x_1 \oplus x_3)$ ,  $f(x_2 \oplus x_3)$  and  $f(x_1 \oplus x_2 \oplus x_3)$  corresponding to odd linear functions containing  $x_3$  are all 0.

*Theorem 3.5.6.* A function is self-dual (anti-self-dual) if and only if its coordinates corresponding to odd linear functions of even (odd) lengths are all 0.

*Proof.* Obvious.

Thus, for example, the three-variable function  $0\ 222\ 000\ \bar{2}$  is self-dual and the three-variable function  $2\ 000\ 222\ 0$  is anti-self-dual.

Next, we shall explain the methods for finding the symmetry group of a Boolean function with a few examples.

*Example 3.5.2.* Find the symmetry group of the four-variable function:

$$\begin{array}{cccccccc}
 & & & & & & & 1 \\
 & & & & & & 1 & 1 & 1 & 2 & 2 \\
 & & & & 1 & 1 & 1 & 2 & 2 & 3 & 3 \\
 0 & 1 & 2 & 3 & 4 & 2 & 3 & 4 & 3 & 4 & 4 \\
 f: & 1 & 3 & 3 & 1 & 1 & \bar{3} & 3 & \bar{1} & \bar{1} & \bar{1} & 1 & 1 & \bar{3} & \bar{1} & 3 & 1
 \end{array}$$

*Solution.* Examining the four coordinates in the second set, we see that possible elements of the symmetry group may be (12), (34) or (12)(34), because only these make the four coordinates invariant. Again examining the four coordinates in the fourth set, we see that possible elements of the symmetry group may be (13), (24) or (13)(24), because the four coordinates taken in the reverse order behave like the four in the second set under permutation operators. But the above two sets of possible elements have no common element. Therefore  $f$  must be perfectly asymmetric.

*Example 3.5.3.* Find the symmetry group of the four-variable function:

$$\begin{array}{cccccccc}
 & & & & & & & 1 \\
 & & & & & & 1 & 1 & 1 & 2 & 2 \\
 & & & & 1 & 1 & 1 & 2 & 2 & 3 & 3 \\
 0 & 1 & 2 & 3 & 4 & 2 & 3 & 4 & 3 & 4 & 4 \\
 f: & 2 & 2 & \bar{2} & \bar{2} & 2 & 2 & 2 & 2 & 2 & \bar{2} & 2 & \bar{2} & 2 & \bar{2} & \bar{2}
 \end{array}$$

*Solution.* Transforming  $f$  by the reflexion operator [23], we obtain

$$[23]f: 2\ 2\ 2\ 2\ 2\ \bar{2}\ \bar{2}\ 2\ 2\ \bar{2}\ 2\ 2\ 2\ \bar{2}\ \bar{2}\ \bar{2}.$$

Then, examining the four coordinates of the fourth set, we see that only (12), (34) and (12)(34) are the possible elements of the symmetry group of  $[23]f$ . Evidently, it is sufficient to consider the six coordinates of the third set in order to test these elements. Thus, transforming the six coordinates by (12), (34) and (12)(34), we obtain  $\bar{2}2\bar{2}2\bar{2}2$ ,  $\bar{2}2\bar{2}2\bar{2}2$  and  $\bar{2}\bar{2}2\bar{2}2\bar{2}2$  respectively. Therefore

$(12)(34)$  is the only one non-trivial element of the symmetry group of  $[23]f$ . Accordingly,  $f$  has only one non-trivial symmetry element and it is given by

$$[23](12)(34)[23]^{-1} = [23](12)(34)[23] = [23][14](12)(34) = [1234](12)(34).$$

*Example 3.5.4.* Find the symmetry group of the four-variable function:

$$f: \begin{array}{cccccccccccccccc} & & & & & & & & & & & & & & & & 1 \\ & & & & & & & & & & & & & & 1 & 1 & 1 & 2 & 2 \\ & & & & & & & & & & & & & 1 & 1 & 1 & 2 & 2 & 3 \\ 0 & 1 & 2 & 3 & 4 & 2 & 3 & 4 & 3 & 4 & 4 & 3 & 4 & 4 & 4 & 4 & 3 & 4 & 4 & 4 & 4 \end{array}$$

*Solution.* From the four coordinates of the second set, we see that possible elements of the symmetry group of  $f$  are those which can be generated from  $(12)$ ,  $(34)$ ,  $[3]$  and  $[4]$ . Thus, applying these elements tentatively to the four coordinates of the fourth set, we obtain  $(12)$ :  $00\bar{4}4$ ,  $(34)$ :  $004\bar{4}$ ,  $[3]$ :  $00\bar{4}4$ ,  $[4]$ :  $00\bar{4}4$ . Hence  $(34)$  is an element of the symmetry group but the others are not. The remaining task is to test  $[34]$ ,  $[3](12)$  and  $[4](12)$ . Thus, transforming the four coordinates by them, we find that all of them are really the elements of the symmetry group. Therefore the symmetry group is the one generated by the three elements  $(34)$ ,  $[34]$  and  $[3](12)$ . It has eight elements and they are: Identity,  $(34)$ ,  $[34]$ ,  $[3](12)$ ,  $[4](12)$ ,  $[34](34)$ ,  $[4](12)(34)$  and  $[3](12)(34)$ .

### 3.6. Coordinate Representation of Functional Separability

Consider a function  $f$  which is functionally separable with respect to variables  $x_1^*$ ,  $x_2^*$ ,  $\dots$ ,  $x_k^*$ , and assume  $f = f^*(g(x_1^*, x_2^*, \dots, x_k^*), x_{k+1}^*, \dots, x_n^*)$  where  $[x_1^*, x_2^*, \dots, x_n^*]$  is a permutation of  $[x_1, x_2, \dots, x_n]$ . Then, expanding  $f$  by  $g$ ,  $f$  can be written in the form:

$$f = g(x_1^*, \dots, x_k^*)\phi(x_{k+1}^*, \dots, x_n^*) + g'(x_1^*, \dots, x_k^*)\phi(x_{k+1}^*, \dots, x_n^*).$$

Now, let  $L$  and  $L'$  be the groups of odd linear functions determined by the bases  $[x_1^*, \dots, x_k^*]$  and  $[x_{k+1}^*, \dots, x_n^*]$  respectively. Then every odd linear function  $y_\lambda$  is expressed uniquely in the form of  $y_\lambda = y_\mu \oplus y_\nu$  where  $y_\mu \in L$  and  $y_\nu \in L'$ . With these preliminaries, we are going to calculate the coordinates  $f(y_\lambda)$ .

To begin with, we have

$$(g \oplus \phi)(y_\lambda) = \sum_{y_\sigma \in L_n} g(y_\mu \oplus y_\sigma)\phi(y_\nu \oplus y_\sigma)/2^{n-1}$$

from (3) of Theorem 3.3.2. But  $g(y_\mu \oplus y_\sigma) = 0$  for every  $y_\sigma \in L$ , because  $g$  is independent of variables  $x_{k+1}^*, \dots, x_n^*$ . Similarly,  $\phi(y_\nu \oplus y_\sigma) = 0$  for every  $y_\sigma \in L'$ . Therefore only the term for  $y_\sigma \in L$  and  $y_\sigma \in L'$ , i.e.,  $y_\sigma = 0$  survives in the above expression, and we obtain

$$(g \oplus \phi)(y_\lambda) = g(y_\mu)\phi(y_\nu)/2^{n-1},$$

which, when used in (2) of Theorem 3.3.2, yields

$$(g\phi)(y_\lambda) = (2^{n-1}\delta_{\lambda 0} + \delta_{\nu 0}g(y_\mu) + \delta_{\mu 0}\phi(y_\nu) - g(y_\mu)\phi(y_\nu)/2^{n-1})/2,$$

because  $g(y_\lambda) = \delta_{\nu 0}g(y_\mu)$  and  $\phi(y_\lambda) = \delta_{\mu 0}\phi(y_\nu)$ .

Applying the same reasoning to  $g'\phi$  and taking account of  $g'(y_\lambda) = -g(y_\lambda)$ , we obtain

$$(g'\phi)(y_\lambda) = (2^{n-1}\delta_{\lambda 0} - \delta_{\nu 0}g(y_\mu) + \delta_{\mu 0}\phi(y_\nu) + g(y_\mu)\phi(y_\nu)/2^{n-1})/2.$$

But  $g\phi \cdot g'\phi = 0$ . Hence, by (1) of Theorem 3.3.3,  $f(y_\lambda)$  is given by

$$f(y_\lambda) = \delta_{\mu 0}(\phi(y_\nu) + \psi(y_\nu))/2 - g(y_\mu)(\phi(y_\nu) - \psi(y_\nu))/2^n, \quad (3.6.1)$$

where  $y_\lambda = y_\mu \oplus y_\nu$ ,  $y_\mu \in L$  and  $y_\nu \in L'$ .

Now, the form of (3.6.1) suggests the following theorem concerning the coordinate representation of functional separability of Boolean functions.

*Theorem 3.6.1. A function  $f$  is functionally separable if and only if there are a partition  $[[x_1^*, \dots, x_k^*], [x_{k+1}^*, \dots, x_n^*]]$  of the set  $[x_1, x_2, \dots, x_n]$  and a function  $g(x_1^*, \dots, x_k^*)$  such that  $2 \leq k \leq n-1$  and*

$$f(y_\lambda) = \alpha(y_\nu)g(y_\mu) \quad (3.6.2)$$

for every  $y_\mu \neq 0$  in  $L$  and every  $y_\nu$  in  $L'$ , where  $y_\lambda = y_\mu \oplus y_\nu$ , and  $L$  and  $L'$  are the group of odd linear functions determined by the bases  $[x_1^*, \dots, x_k^*]$  and  $[x_{k+1}^*, \dots, x_n^*]$  respectively.

*Proof.* The "only if" part has been already proved because (3.6.1) becomes

$$f(y_\lambda) = -g(y_\mu)(\phi(y_\nu) - \psi(y_\nu))/2^n$$

for  $y_\mu \neq 0$ . The "if" part is proved as follows.

Let us put

$$\beta(y_\nu) = f(y_\nu) - \alpha(y_\nu)g(0),$$

then  $f(y_\lambda)$  is written as  $f(y_\lambda) = \delta_{\mu 0}\beta(y_\nu) + \alpha(y_\nu)g(y_\mu)$ .

Now, from (3.4.12), we have

$$\begin{aligned} \pm 2^{n-1} &= \sum_{\lambda} \delta_{\lambda} f(y_{\lambda}) = \sum_{\mu} \sum_{\nu} \delta_{\mu} \delta_{\nu} (\delta_{\mu 0} \beta(y_{\nu}) + \alpha(y_{\nu}) g(y_{\mu})) \\ &= \sum_{\nu} \delta_{\nu} \beta(y_{\nu}) + \sum_{\mu} \delta_{\mu} g(y_{\mu}) \sum_{\nu} \delta_{\nu} \alpha(y_{\nu}). \end{aligned}$$

But  $\sum_{\mu} \delta_{\mu} g(y_{\mu}) = \pm 2^{n-1}$  because  $g$  is independent of  $x_{k+1}^*, \dots, x_n^*$ . Furthermore the two values  $2^{n-1}$  and  $-2^{n-1}$  can be actually attained by suitable choices of the values of  $\delta_{\mu}$ , because  $g$  is neither 0 nor 1. Thus, we have

$$\sum_{\nu} \delta_{\nu} (\beta(y_{\nu}) \pm 2^{n-1} \alpha(y_{\nu})) = \pm 2^{n-1}.$$

Hence, by Theorem 3.4.6 and Theorem 3.5.5,  $\beta(y_{\nu}) + 2^{n-1} \alpha(y_{\nu})$  and  $\beta(y_{\nu}) - 2^{n-1} \alpha(y_{\nu})$  are the coordinates of two functions which are independent of  $x_1^*$ ,

$\dots, x_k^*$ . Let us denote these two functions by  $\psi$  and  $\phi$  respectively. Then we obtain

$$\beta(y_v) + 2^{n-1}\alpha(y_v) = \psi(y_v) \quad \text{and} \quad \beta(y_v) - 2^{n-1}\alpha(y_v) = \phi(y_v),$$

and consequently,

$$\alpha(y_v) = (\psi(y_v) - \phi(y_v))/2^n \quad \text{and} \quad \beta(y_v) = (\psi(y_v) + \phi(y_v))/2.$$

Therefore  $f$  must be of the form

$$f(y_\lambda) = \delta_{\mu 0}(\phi(y_v) + \psi(y_v))/2 - (\phi(y_v) - \psi(y_v))g(y_\mu)/2^n.$$

This completes the proof.

The theorem is theoretically fairly simple, but it requires a large amount of calculations for practical applications, because we must test the condition (3.6.2) for every possible partition of the set of variables. However, when the symmetry structures of functions are taken into account, the required work will be reduced considerably under most circumstances. In this respect, the viewpoints of Section 2.4, especially, Theorem 2.4.3 and Theorem 2.4.4 are useful.

Now we shall work out a few examples illustrating some typical situations in the test of functional separability.

*Example 3.6.1.* Test the functional separability of the four-variable function:

$$f: 1 \ 3 \ 3 \ 1 \ 1 \ \bar{3} \ 3 \ \bar{1} \ \bar{1} \ 1 \ 1 \ \bar{3} \ \bar{1} \ 3 \ 1.$$

*Solution.* This function is shown to be perfectly asymmetric in Example 3.5.2. Hence it is not functionally separable, because no function of less than four variables is perfectly asymmetric.

*Example 3.6.2.* Test the functional separability of the four-variable function:

$$f: 2 \ 2 \ \bar{2} \ \bar{2} \ 2 \ 2 \ 2 \ 2 \ 2 \ \bar{2} \ 2 \ \bar{2} \ 2 \ \bar{2} \ \bar{2}.$$

*Solution.* It has been shown in Example 3.5.3 that  $f$  has the unique non-trivial symmetry  $[1234](12)(34)$ . Since no function of less than four variables is perfectly asymmetric nor has the unique non-trivial symmetry  $[1234](12)(34)$ ,  $f$  is not functionally separable.

*Example 3.6.3.* Test the functional separability of the four-variable function:

$$f: 0 \ 4 \ 4 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 4 \ \bar{4} \ 0.$$

*Solution.* The symmetry group of  $f$  is:

$$[I, (34), [34], [3](12), [4](12), [34](34), [3](12)(34), [4](12)(34)],$$

where  $I$  is the identity element (see Example 3.5.4). This group has many subgroups among which only  $[I, [3](12)]$ ,  $[I, [4](12)]$  and  $[I, (34), [34], [34](34)]$  can be symmetry groups of functions of less than four variables. Accordingly, if  $f$  is functionally separable, it must be so with respect to one of  $[x_1, x_2, x_3]$ ,

$[x_1, x_2, x_4]$ ,  $[x_3, x_4]$ ,  $[x_1, x_3, x_4]$  or  $[x_2, x_3, x_4]$ . But, by (1) of Theorem 2.4.3, if  $f$  is functionally separable with respect to  $[x_1, x_2, x_3]$  or  $[x_1, x_2, x_4]$ , it is so with respect to  $[x_3, x_4]$ , since  $f$  is invariant under (34). Furthermore, by (3) of Theorem 2.4.3,  $f$  is not functionally separable with respect to  $[x_1, x_3, x_4]$  and  $[x_2, x_3, x_4]$ , since  $f$  is invariant under [3](12).

To sum up: If  $f$  is functionally separable at all, it must be of the form  $f = f^*(g(x_3, x_4), x_1, x_2)$ . Hence it suffices to test the condition (3.6.2) for the partition  $[[x_3, x_4], [x_1, x_2]]$ .

Let us arrange the 16 coordinates of  $f$  as shown in Table 3.6.1. Then it will be found that (3.6.2) can be satisfied by taking  $g(y_\mu)$  and  $\alpha(y_\nu)$  as shown in the bottom row and the sixth column respectively of Table 3.6.1. The values of  $\beta(y_\nu)$ ,  $\phi(y_\nu)$  and  $\psi(y_\nu)$  calculated from  $g(y_\mu)$  and  $\alpha(y_\nu)$  are shown in the seventh, eighth and ninth columns of the table.

Thus, it is found that  $f$  is really functionally separable. Moreover  $f$  is given explicitly as  $f = x_2(x_3 \oplus x_4) + x_1(x_3 \oplus x_4)'$ , since  $g = x_3 \oplus x_4$ ,  $\phi = x_2$  and  $\psi = x_1$ .

TABLE 3.6.1

$y_\mu \rightarrow$ $y_\nu \downarrow$	0	3	4	3	4	$\alpha$	$\beta$	$\phi$	$\psi$
0	0	0	0	0	0	0	0	0	0
1	4	0	0	4		1/2	4	0	8
2	4	0	0	4		-1/2	4	8	0
12	0	0	0	0		0	0	0	0
$g$	0	0	0	8					

### 3.7. Divisible Boolean Functions

A Boolean function  $f$  is said to be *divisible* if and only if  $L(f)$ , the group of odd linear functions  $y_\mu$  such that  $f \leq y_\mu^{\otimes \mu}$ , contains at least one function other than 0.

Now, consider a divisible function  $f$  and let  $[y_1, y_2, \dots, y_k]$  be any base of  $L(f)$ . Further assume  $f \leq y_i^{\otimes i}$  ( $i = 1, 2, \dots, k$ ). Then, by (3.4.2), we have

$$f(y_\mu) = 2^{n-1} \delta_{\mu 0} - \delta_\mu d(f) \quad (3.7.1)$$

for any  $y_\mu \in L(f)$ , where  $\delta_i = -\varepsilon_i$  ( $i = 1, 2, \dots, k$ ) and  $\delta_{\rho \oplus \sigma} = \delta_\rho \delta_\sigma$  for any  $y_\rho$  and  $y_\sigma$  in  $L(f)$ . Next, let any  $n-k$  variables which are not in  $L(f)$  be  $x_1^*, x_2^*, \dots, x_{n-k}^*$  and let the group of odd linear functions determined by the base  $[x_1^*, x_2^*, \dots, x_{n-k}^*]$  be denoted by  $L^*$ . Then any  $y_\lambda$  in  $L_n$  is represented uniquely in the form

$$y_\lambda = y_\mu \oplus y_\nu \quad (3.7.2)$$

where  $y_\mu \in L(f)$  and  $y_\nu \in L^*$ . When  $y_\lambda$  is represented in the above form, (3.4.1) yields

$$2^{n-1} \delta_{\lambda 0} - f(y_\lambda) = \delta_\mu (2^{n-1} \delta_{\nu 0} - f(y_\nu)). \quad (3.7.3)$$

This indicates that the knowledge of  $f(y_\nu)$  for  $y_\nu \in L^*$  is sufficient to determine all the coordinates  $f(y_\lambda)$  for  $y_\lambda \in L(f)$ .

Now, from (3.7.2), (3.7.3) and (3.4.12), we obtain

$$\sum_\lambda \delta'_\lambda f(y_\lambda) = 2^{n-1} - \sum_\mu \delta'_\mu \delta_\mu \sum_\nu \delta'_\nu (2^{n-1} \delta_{\nu 0} - f(y_\nu)) = \pm 2^{n-1}$$

because  $\delta'_\lambda = \delta'_\mu \delta'_\nu$ . If we take  $\delta'_\mu = \delta_\mu$  for every  $y_\mu \in L(f)$ , the above equation becomes

$$2^{n-1} - 2^k \sum_\nu \delta'_\nu (2^{n-1} \delta_{\nu 0} - f(y_\nu)) = \pm 2^{n-1}.$$

Here, let us put

$$2^{n-1} \delta_{\nu 0} - f(y_\nu) = 2^{n-k-1} \delta_{\nu 0} - f^*(y_\nu), \quad (3.7.4)$$

then we obtain

$$\sum_\nu \delta'_\nu f^*(y_\nu) = \pm 2^{n-k-1},$$

which means that  $[f^*(y_\nu)]$  ( $y_\nu \in L^*$ ) represents a Boolean function  $f^*$  of  $n-k$  variables  $x_1^*, x_2^*, \dots, x_{n-k}^*$ . Clearly  $f^*$  has the same dimension as  $f$  and is indivisible.

Now we shall show that  $f = y_1^{\varepsilon_1} y_2^{\varepsilon_2} \cdots y_k^{\varepsilon_k} f^*$ . First we note that  $f^*$ , when considered as a degenerate function of  $x_1, x_2, \dots, x_n$ , has the coordinates

$$f^*(y_\lambda) = \delta_{\mu 0} 2^k f^*(y_\nu),$$

because

$$\sum_\lambda \delta'_\lambda f^*(y_\lambda) = 2^k \sum_\nu \delta'_\nu f^*(y_\nu) = \pm 2^{n-1}.$$

Further, by Theorem 3.4.2,  $g = y_1^{\varepsilon_1} y_2^{\varepsilon_2} \cdots y_k^{\varepsilon_k}$  has the coordinates

$$g(y_\lambda) = \delta_{\nu 0} (2^{n-1} \delta_{\mu 0} - 2^{n-k} \delta_\mu).$$

Therefore, by (3) of Theorem 3.3.2, we obtain

$$\begin{aligned} (g \oplus f^*)(y_\lambda) &= \sum_\rho g(y_\mu \oplus y_\rho) f^*(y_\rho \oplus y_\nu) / 2^{n-1} = (2^{n-1} \delta_{\mu 0} - 2^{n-k} \delta_\mu) 2^k f^*(y_\nu) / 2^{n-1} \\ &= (2^k \delta_{\mu 0} - 2 \delta_\mu) f^*(y_\nu) = f^*(y_\lambda) - 2 \delta_\mu f^*(y_\nu). \end{aligned}$$

Then, it follows, by (2) of Theorem 3.3.2, that

$$\begin{aligned} (gf^*)(y_\lambda) &= (2^{n-1} \delta_{\lambda 0} + \delta_{\nu 0} (2^{n-1} \delta_{\mu 0} - 2^{n-k} \delta_\mu) + f^*(y_\lambda) - (f^*(y_\lambda) - 2 \delta_\mu f^*(y_\nu))) / 2 \\ &= 2^{n-1} \delta_{\lambda 0} - \delta_\mu (2^{n-k-1} \delta_{\nu 0} - f^*(y_\nu)), \end{aligned}$$

because  $\delta_{\lambda 0} = \delta_{\mu 0} \delta_{\nu 0}$ . Since it is seen that  $(gf^*)(y_\lambda) = f(y_\lambda)$  by (3.7.3) and (3.7.4),  $f = y_1^{\varepsilon_1} y_2^{\varepsilon_2} \cdots y_k^{\varepsilon_k} f^*$  is proved.

In dealing with divisible functions, it is convenient to assume that they are of the *standard form*, where a divisible function  $f$  is said to be of the standard form if and only if  $f \leq y_i^1$  for every odd linear function  $y_i$  of a base of the group  $L(f)$ . Thus, for example, the functions

$$6 \quad \bar{2} \quad \bar{2} \quad \bar{2} \quad 0 \quad \bar{2} \quad \bar{2} \quad 0 \quad \bar{2} \quad 0 \quad 0 \quad \bar{2} \quad 0 \quad 0 \quad 0,$$

and

$$5 \quad 1 \quad 1 \quad 1 \quad 1 \quad \bar{3} \quad \bar{3} \quad 1 \quad \bar{3} \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1$$

are of the standard form. For divisible functions of the standard form, (3.7.1) and (3.7.3) are simplified as

$$f(y_\lambda) = 2^{n-1} \delta_{\lambda 0} - d(f) \quad (y_\lambda \in L(f)) \quad (3.7.1')$$

$$\text{and} \quad 2^{n-1} \delta_{\lambda 0} - f(y_\lambda) = 2^{n-1} \delta_{\nu 0} - f(y_\nu) \quad (y_\lambda = y_\mu \oplus y_\nu, y_\mu \in L(f), y_\nu \in L^*) \quad (3.7.3')$$

respectively.

Note that the definition of the standard form of divisible functions is compatible with the previous one of functions of odd dimensions given in Section 3.5. Precisely speaking, if a divisible function of an odd dimension is of the standard form in the previous sense, it is of the standard form in the present sense, too.

Now we shall prove that every divisible function can be transformed into the standard form by a suitable reflexion operator. In order to transform a divisible function  $f$  such that  $f \leq y_i^{\varepsilon_i}$  ( $i = 1, 2, \dots, k$ ) for odd linear functions  $y_i$  of a base of  $L(f)$  into the standard form, we need a reflexion operator which transform  $[y_1, y_2, \dots, y_k]$  into  $[y_1^{-\varepsilon_1}, y_2^{-\varepsilon_2}, \dots, y_k^{-\varepsilon_k}]$ . The existence of such a reflexion operator is shown in the following theorem.

*Theorem 3.7.1. Let  $[y_1, y_2, \dots, y_n]$  be any base of  $L_n$ , then there is a unique reflexion operator which transform  $[y_1, y_2, \dots, y_n]$  into  $[y_1^{\varepsilon_1}, y_2^{\varepsilon_2}, \dots, y_n^{\varepsilon_n}]$  for every combination of values of  $\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n$ .*

*Proof.* Associate with each atom  $x_1^{\varepsilon_1}, x_2^{\varepsilon_2} \cdots x_n^{\varepsilon_n}$  the reflexion operator priming only the variables  $x_i$  such that  $\varepsilon_i = 1$ . Then there arises a one-to-one correspondence between atoms and reflexion operators. In this correspondence, the reflexion operators which transform an odd linear function  $y$  into  $y'$  correspond to atoms  $a$  such that  $a \leq y$  (see the proof Theorem 2.2.1). Therefore the reflexion operator in question is given by that which corresponds to the atom  $y_1^{-\varepsilon_1}, y_2^{-\varepsilon_2} \cdots y_n^{-\varepsilon_n}$ . This completes the proof.

### 3.8. Linear Automorphisms and Linear Transformations

In Section 3.4, we have seen that, for any set of  $n$  mutually independent odd linear functions  $[y_1, y_2, \dots, y_n]$ , every function of the form  $y_1^{\varepsilon_1} y_2^{\varepsilon_2} \cdots y_n^{\varepsilon_n}$  is an atom, and conversely every atom is expressed in this form. Hence the mapping  $\tau: f(x_1, x_2, \dots, x_n) \rightarrow f(y_1^{\varepsilon_1}, y_2^{\varepsilon_2}, \dots, y_n^{\varepsilon_n})$  induces a permutation of atoms, and therefore, by Theorem 2.3.1, it is an automorphism of  $B_n$ . Moreover, it is an automorphism of  $L_n^*$ , the group of all linear functions (see Section 2.2). In this sense, the mapping is called a *linear automorphism* (of  $B_n$ ). Obviously, a linear automorphism is a symmetry if and only if it preserves the length of the linear function.

The effects of a linear automorphism  $\tau$  on the coordinates of Boolean functions are similar to those of a symmetry. That is: For any function  $f$ , we have

$$(\tau f)(y_\lambda) = f(\tau^{-1} y_\lambda). \quad (3.8.1)$$

Therefore the effects are a permutation and/or changes of signs of coordinates. Precisely, when  $\tau: x_i \rightarrow y_i^{\varepsilon_i}$  is divided into two steps,  $x_i \rightarrow y_i$  and  $y_i \rightarrow y_i^{\varepsilon_i}$ , the effect of the first step is a permutation and that of the second step is changes of signs of coordinates.

*Example 3.8.1.* Apply the linear automorphism  $\tau: x_1 \rightarrow x_1 \oplus x_3, x_2 \rightarrow x_2 \oplus x_3, x_3 \rightarrow x'_3$  to the function:

$$f: \begin{array}{ccccccc} & & & & & & 1 \\ & & & & 1 & 1 & 2 & 2 \\ 0 & 1 & 2 & 3 & 2 & 3 & 3 & 3 \\ f: & 1 & 3 & 1 & 1 & \bar{1} & \bar{1} & 1 & \bar{1}. \end{array}$$

*Solution.*  $\tau^{-1}(x_1 \oplus x_3) = x_1, \tau^{-1}(x_2 \oplus x_3) = x_2, \tau^{-1}x_3 = x'_3.$

Solving these equations, we have

$$\tau^{-1}x_1 = (x_1 \oplus x_3)', \tau^{-1}x_2 = (x_2 \oplus x_3)', \tau^{-1}x_3 = x'_3,$$

whence the coordinates of  $\tau f$  is given by:

$$\tau f: 1 \ 1 \ \bar{1} \ \bar{1} \ \bar{1} \ 3 \ 1 \ 1.$$

There is one more kind of mappings with the similar effects on the coordinates of Boolean functions. Such mappings are induced by the ring addition of linear functions  $y_\mu^e$ . For the sake of brevity, we call these mappings  $f \rightarrow f \oplus y_\mu^e$  as *linear transformations*.

Since we have  $(f \oplus y_\mu^e)(y_\lambda) = \epsilon f(y_\lambda \oplus y_\mu)$  (3.8.2)

from (3.3.3) and (3) of Theorem 3.3.2, the effects of a linear transformation are a permutation of the coordinates and/or the change of signs of all the coordinates. Although the linear automorphism and the linear transformation are very alike in regard to the effects on the coordinates of Boolean functions, they are entirely different kinds of mappings. In fact, a linear transformation is never an automorphism, because 0 is not invariant under a linear transformation.

Now, in view of the simplicity of the effects of the linear automorphisms and the linear transformations on the coordinates of Boolean functions, we introduce the following four definitions.

Two functions are said to be *homologous (analogous)* if and only if one of them is transformed into another by a linear automorphism (a linear transformation).

Two functions are said to be of the same *prototype (family)* if and only if one of them is transformed into another by a linear automorphism (symmetry) and/or a linear transformation.

Thus, we have four concepts of type, genus, family and prototype for the classification of Boolean functions. Comparing their definitions, it will be seen that they form an increasing sequence in regard to the generality in the above listed order.

The concept of prototype is the most general. Indeed it is so general that Boolean functions may be classified into a surprisingly small number of prototypes. For example, as will be seen in the next chapter, 65,536 Boolean functions of four variables are classified into only eight prototypes. Next comes the



concept of family. It is an intermediate concept between prototype and genus, and is significant in the sense of the following theorem.

*Theorem 3.8.1. Functions which are analogous to a function  $f$  and functions which are analogous to another function  $g$  are classified into the same set of genera, if and only if  $f$  and  $g$  are of the same family.*

*Proof. "If" part:* If  $f$  and  $g$  are of the same family, then there are a linear function  $y_\lambda^\delta$  and a symmetry  $\sigma$  such that  $\sigma(f \oplus y_\lambda^\delta) = g$ . Therefore, for each linear function  $y_\mu^\varepsilon$ , we have

$$\sigma(f \oplus (y_\lambda^\delta \oplus \sigma^{-1} y_\mu^\varepsilon)) = g \oplus y_\mu^\varepsilon$$

and

$$f \oplus y_\mu^\varepsilon = \sigma^{-1}(g \oplus \sigma(y_\lambda^\delta \oplus y_\mu^\varepsilon)).$$

It follows then that each function which is analogous to  $g$  is congruent with a certain function which is analogous to  $f$  and vice versa. Thus, the "if" part is proved.

*"Only if" part:* If we assume the conclusion of the theorem, then, for each linear function  $y_\lambda^\delta$ , there are a linear function  $y_\mu^\varepsilon$  and a symmetry  $\sigma$  such that  $\sigma(f \oplus y_\lambda^\delta) = g \oplus y_\mu^\varepsilon$ , i.e.,  $\sigma(f \oplus (y_\lambda^\delta \oplus \sigma^{-1} y_\mu^\varepsilon)) = g$ . But this means that  $f$  and  $g$  are of the same family. Thus, the "only if" part is proved.

The theorem tells us that, in order to classify Boolean functions into genera, it is of advantage to classify them into families first, since every genus will then be represented by a function analogous to an arbitrary representative of a unique family. In the next chapter, this plan will be actually used for the classification of Boolean functions of three and four variables.

### 3.9. Geometric Derivation of Coordinate Representation

The coordinate representation has been derived algebraically through the matrix representations. But it is also possible to derive the coordinate representation geometrically. As a matter of fact, the original work of D. E. Muller<sup>1)</sup> was along this line. We shall now describe Muller's geometric derivation.

As we have seen in Section 2.6.2, a Boolean function of  $n$  variables is uniquely represented by a vertex of a  $2^n$ -cube. Let us take a new coordinate system, say, the C-system with the origin at the center of the cube and the coordinate axes which are parallel to the sides of the cube. Referred to the C-system, a vertex corresponding to an arbitrary function  $f$  has the coordinates  $(\delta_0/2, \delta_1/2, \dots, \delta_{2^n-1}/2)$ , where  $\delta_i = 1$  if  $f \geq a_i$  and  $\delta_i = -1$  if  $f \not\geq a_i$ . The Euclidean distance between any vertex and the center is  $2^{n/2-1}$ . Let the inner product of the radius vectors for the vertices  $f$  and  $g$  be denoted by  $(f, g)$  as usual, then we have

$$(f, g) = (g, f) = \sum_i \delta_i \varepsilon_i / 4,$$

where we assume that the vertex  $g$  has the coordinates  $(\varepsilon_0/2, \varepsilon_1/2, \dots)$ . Now observe:  $\delta_i \varepsilon_i = 1$  if  $a_i \leq (f \oplus g)'$  and  $\delta_i \varepsilon_i = -1$  if  $a_i \leq f \oplus g$ . Hence we have

<sup>1)</sup> cf. Ref. 20.

$$(f, g) = (d((f \oplus g)') - d(f \oplus g))/4. \quad (3.9.1)$$

and, in particular,

$$(0, f) = (d(f') - d(f))/4. \quad (3.9.2)$$

Note here that (3.9.2) implies

$$(0, f \oplus g) = (d((f \oplus g)') - d(f \oplus g))/4.$$

Therefore, from (3.9.1), we have

$$(f, g) = (0, f \oplus g) \quad (3.9.3)$$

and more generally

$$(f \oplus h, g) = (f, g \oplus h), \quad (3.9.4)$$

because both sides of (3.9.4) are equal to  $(0, f \oplus g \oplus h)$  by (3.9.3).

Now, since the dimension of every linear function other than 0 and 1 is  $2^{n-1}$ , we obtain

$$(0, y_\lambda^{\delta}) = \delta 2^{n-2} \delta_{\lambda 0} \quad (3.9.5)$$

from (3.9.1), and

$$(y_\lambda^{\delta}, y_\mu^{\varepsilon}) = \delta \varepsilon 2^{n-2} \delta_{\lambda \mu} \quad (3.9.6)$$

from (3.9.3). Thus, the following theorem is established.

*Theorem 3.9.1.  $2^n$  lines connecting the center of  $2^n$ -cube and vertices corresponding to linear functions are orthogonal to one another.*

The theorem suggests the advantage, in regard to the representation of Boolean functions, of using a new coordinate system having the origin at the center of the cube and the axes passing the vertices corresponding to the linear functions. It is evidently rectangular. Let us call it *L-system* because it is defined by the linear functions. The positive direction of each axis of the *L-system* may be taken as the direction from the origin to the vertex corresponding to the odd linear function passed by it. The coordinate transformation from *C-system* to *L-system* is given by the orthogonal matrix:

$$S = 2^{-n/2+1}(\xi_{\lambda i}) \quad (\lambda, i = 0, 1, \dots, 2^n - 1), \quad (3.9.7)$$

where  $(\xi_{\lambda i})$  ( $i = 0, 1, \dots, 2^n - 1$ ) is the row vector having the coordinates of odd linear function  $y_\lambda$  in *C-system* as the components.

Now let us define: By the *coordinates of a Boolean function*  $f$  we mean the coordinates of the corresponding vertex in *L-system* multiplied by  $2^{n/2}$ . Then it will be observed that the coordinate  $f(y_\lambda)$  is given by

$$f(y_\lambda) = 2(y_\lambda, f). \quad (3.9.8)$$

Hence, from (3.9.1) and (3.9.2), we obtain

$$f(0) = 2^{n-1} - d(f) \quad (3.9.9)$$

and

$$f(y_\lambda) = d(f y_\lambda) - d(f y'_\lambda) \quad (3.9.9')$$

for every  $\lambda \neq 0$ .

The coordinate representation thus derived geometrically is the same as that derived algebraically, because (3.9.9) and (3.9.9') agree with (3.3.2) and (3.3.2') respectively.

#### 4. Classification of Boolean Functions of Three and Four Variables

##### 4.1. Significance of Classification of Boolean Function

The exact meaning of classification of Boolean functions is to find a class of Boolean functions which is complete in the sense that, for any given function, there is only one function which is congruent with it.

Congruent functions are very alike in almost every respect. Their dimensions are equal; they have similar minimal sums with the same number of essential prime implicants, of essential literals, and of occurrences of literals; They may be all symmetrizable, functionally separable, divisible etc. or all not. Furthermore, they can be realized by physically similar switching circuits. For example, they can be realized by similar relay contact circuits with the same number of contacts and of transfer contacts. Thus, by classifying Boolean functions and considering only the representatives, a great amount of research efforts can be saved in the study of theory and applications of Boolean functions.

The preliminary stage for the classification of Boolean functions is the enumeration of types of Boolean functions. Accordingly, it is worth while to make a survey of the present state of development in this field.

Slepian's<sup>1)</sup> work is the most important in this field. He devised an ingenious method for this problem and counted the number of types of Boolean functions up to  $n=6$ . His results are reproduced in Table 4.1.1, where  $T_n$  is the number of types of Boolean functions of  $n$  variables.

TABLE 4.1.1

$n$	1	2	3	4	5	6
$T_n$	3	6	22	402	1, 228, 158	400, 507, 806, 843, 728

Historically, the first successful approach to this problem was made by Pólya.<sup>2)</sup> He counted the numbers of types of  $n$ -variable Boolean functions with dimensions  $d$ ,  $T_n^{(d)}$ , for  $d=0, 1, \dots, 2^n$  and  $n=1, 2, 3, 4$ . His results were correct but his method was rather intuitive and short of mathematical rigor. Pólya's results are shown in Table 4.1.2.

TABLE 4.1.2

$n \downarrow d \rightarrow$	0	1	2	3	4	5	6	7	8	$T_n$
1	1	1	1							3
2	1	1	2	1	1					6
3	1	1	3	3	6	3	3	1	1	22
4	1	1	4	6	19	27	50	56	74	402

<sup>1)</sup> cf. Ref. 37. <sup>2)</sup> cf. Ref. 28.

In the table, the values of  $T_n^{(d)}$  for  $d > 8$  are omitted, because they can be obtained by the obvious formula  $T_n^{(d)} = T_n^{(2^n-d)}$ . There are some explicit formulas of  $T_n^{(d)}$  for moderate values of  $d$ , among which  $T_n^{(0)} = 1$  and  $T_n^{(1)} = 1$  are evident, and  $T_n^{(2)} = n$  is almost evident. Concerning  $T_n^{(3)}$ , Durst derived the formula<sup>1)</sup>

$$T_n^{(3)} = \sum_{k=2}^n ([k/3] + \sum_{r=0}^{[k/3]} [(k-3r)/2])$$

by using the distance between atoms, where  $[x]$  is the usual notation for the integral part of a real number  $x$ . Recently the present author<sup>2)</sup> has made an attempt to count  $T_n^{(3)}$  and  $T_n^{(4)}$  by means of the coordinate representation. He has not only reproduced Durst's result for  $T_n^{(3)}$  but also succeeded in deriving an explicit formula for  $T_n^{(4)}$ . However the formula is too complicated to be presented here. So we shall only show the values of  $T_n^{(4)}$  together with those of  $T_n^{(3)}$  calculated for moderate  $n$ , in the following table.

TABLE 4.1.3

$n$	1	2	3	4	5	6	7	8	9	10
$T_n^{(3)}$	0	1	3	6	10	16	23	32	43	56
$T_n^{(4)}$	0	1	6	19	47	103	203	373	649	1079

On the other hand, the number of genera of Boolean functions of  $n$  variables,  $G_n$  has been calculated recently by the author<sup>3)</sup> up to  $n=6$ . The results are shown in Table 4.1.4, where  $S_n$  is the number of genera (types) of self-complementary Boolean functions of  $n$  variables. Obviously, we have  $G_n = (T_n + S_n)/2$ . The table contains also the values of  $T_n^{(2^n-1)}$  obtained from Table 4.1.2. By comparing  $S_n$  and  $T_n^{(2^n-1)}$ , we see that every function of the dimension  $2^{n-1}$  is self-complementary for  $n \leq 3$ , but the same is not true for  $n \geq 4$ .

TABLE 4.1.4

$n$	1	2	3	4	5	6
$G_n$	2	4	14	222	616, 126	400, 253, 952, 527, 184
$S_n$	1	2	6	42	4, 094	98, 210, 640
$T_n^{(2^n-1)}$	1	2	6	74	*	*

Turning to Table 4.1.1, it is noted that  $T_n$  are already very large even for moderate values of  $n$ . In this extraordinary growth of  $T_n$ , we see a practical limitation for the possibility of classification of Boolean functions. Indeed, no attempt has been made so far beyond  $n=4$ . Even for  $n=4$ , the problem involves serious difficulties. There seems to be no practical solution for this problem other than the following one. That is: "Choose an arbitrary function. Apply every possible symmetry to it and record all different functions obtained. Choose another arbitrary function which is not recorded in the previous step. Apply

<sup>1)</sup> cf. Ref. 7. <sup>2)</sup> cf. Ref. 25 (unpublished). <sup>3)</sup> cf. Ref. 24.

every possible symmetry to it and record all different functions obtained. Continue the process until all the functions are exhausted". As a matter of fact, the only one contribution, so far as the author is aware of, for  $n=4$  by the Staff of the Computation Laboratory of Harvard University has been carried out by this method with a high speed computer.<sup>1)</sup>

The author's intention here is to attack the same problem by an entirely different method based on the coordinate representation. The development in the following sections will show that the intention proves to be fairly successful in reducing the process of classification largely and in finding out some new aspects of classification which have been hitherto overlooked.

#### 4. 2. Principle of the Method of Classification

The classification will be carried out in four stages from the wider to the narrower in accordance with the four concepts: prototype, family, genus and type. Before proceeding to the description of the four stages, the definitions of the four concepts will be relisted for the sake of definiteness.

"Two Boolean functions are of the same prototype if and only if one of them is transformed into another by a linear automorphism and/or a linear transformation".

"Two Boolean functions are of the same family if and only if one of them is transformed into another by a symmetry and/or a linear transformation".

"Two Boolean functions are of the same genus if and only if one of them is transformed into another by a symmetry and/or the complementation".

"Two Boolean functions are of the same type if and only if one of them is transformed into another by a symmetry".

##### *Stage 1: Classification into Prototypes*

In the beginning, Boolean functions will be classified according to the greatest absolute value of the coordinates. For simplicity, the above mentioned value is called *index*. Then each class of functions with the same index will be easily classified into prototypes by the theory of Chapter 3. In doing this, we may only consider those functions the first coordinates of which are equal to the indices, because any function can be transformed into this form, if necessary, by a suitable linear transformation. Each prototype obtained in this stage will be represented by an appropriately chosen function of the above form.

##### *Stage 2: Classification into Families*

In this stage, each prototype obtained in Stage 1 will be classified into families. Practically, this will be done indirectly as follows. That is: First, functions which are homologous to the representative of the prototype will be classified into types. Then, it will be examined whether the types obtained are of different families or not. In order to do this, some calculations are necessary but no serious difficulties are involved. At any rate, in the end of Stage 2, we have the list of representatives of all the families.

##### *Stage 3: Classification into Genera*

In this stage, each family obtained in Stage 2 will be classified into genera.

<sup>1)</sup> cf. Ref. 38.

This can be done by applying all linear transformations to the representative of the family, since, as we have seen in Section 3.8, every genus of the family is represented by a function which is analogous to the representative. It is not true, however, that every genus should be represented by only one of such functions. Therefore some functions are redundant as the representative of the genera and must be eliminated. For this purpose, a closer investigation concerning the set of functions which are analogous to the representative of a family is necessary.

The most apparent fact in this respect is that  $(f \oplus y_\lambda)' = f \oplus y'_\lambda$ , where  $f$  is the representative of a family and  $y_\lambda$  is an odd linear function. Therefore either of  $f \oplus y_\lambda$  or  $f \oplus y'_\lambda$  is redundant for each  $y_\lambda$ . Here, as a possible decision, we take  $f \oplus y_\lambda$  and discard  $f \oplus y'_\lambda$  for each  $y_\lambda$ . Thus, 2" redundant functions of the form  $f \oplus y'_\lambda$  have been eliminated. Further eliminations are still necessary and the following lemma serves for this purpose.

*Lemma 4.2.1. Let  $f$  be any function, and let  $y_\lambda$  and  $y_\mu$  be any odd linear functions, then  $f \oplus y_\lambda$  and  $f \oplus y_\mu$  are of the same genus if and only if either  $\sigma y_\lambda = y_\mu^\varepsilon$  for a symmetry  $\sigma$  such that  $\sigma f = f^\delta$ , or  $\sigma(y_\lambda \oplus y_\nu) = y_\mu^\varepsilon$  for a symmetry  $\sigma$  and an odd linear function  $y_\nu$  such that  $\sigma(f \oplus y_\nu) = f^\delta$ .*

*Proof.* The first condition being the special case of the second condition for  $y_\nu = 0$ , it suffices to consider only the second condition.

*"If" part:* Forming the ring sum of  $\sigma(y_\lambda \oplus y_\nu) = y_\mu^\varepsilon$  and  $\sigma(f \oplus y_\nu) = f^\delta$ , we obtain  $\sigma(f \oplus y_\lambda) = (f \oplus y_\mu)^{\delta\varepsilon}$  which means that  $f \oplus y_\lambda$  and  $f \oplus y_\mu$  are of the same genus.

*"Only if" part:* Assume that  $f \oplus y_\lambda$  and  $f \oplus y_\mu$  are of the same genus, then there is a symmetry  $\sigma$  such that  $\sigma(f \oplus y_\lambda) = (f \oplus y_\mu)^{\varepsilon'}$ , i.e.,  $\sigma f \oplus f = \sigma y_\lambda \oplus y_\mu^{\varepsilon'}$ . Since the right hand side of the last equation is a linear function, say,  $\sigma y_\nu^\delta$ , we obtain  $\sigma(f \oplus y_\nu) = f^\delta$  and  $\sigma(y_\lambda \oplus y_\nu) = y_\mu^\varepsilon$  where  $\varepsilon = \sigma\varepsilon'$ .

Now, by the lemma, the elimination of the redundant functions will be carried out as follows. That is: Classify, for each representative  $f$ , all the odd linear functions into classes in accordance with the conditions of Lemma 4.2.1, then functions  $f \oplus y_\lambda$  for all but one  $y_\lambda$  in each class are redundant and can be eliminated.

First, only the first condition of the lemma will be taken into account and any two odd linear functions  $y_\lambda$  and  $y_\mu$  will be put into the same class if and only if  $\sigma y_\lambda = y_\mu$  for a symmetry  $\sigma$  such that  $\sigma f = f^\delta$ . Here, we notice the following two facts. The one is that we may neglect the possibility  $\sigma f = f'$ , since the first coordinate  $f(0)$  is not 0, being equal to the index, and therefore,  $f$  is not self-complementary. The other is that, for any permutation operator  $\pi$ , any reflexion operator  $\gamma$  and any odd linear function  $y_\lambda$ ,  $\pi y_\lambda$  is an odd linear function and  $\gamma y_\lambda$  is either  $y_\lambda$  or  $y'_\lambda$ . Taking these facts into account, we may proceed as follows. That is: Put any  $y_\lambda$  and  $y_\mu$  such that  $\pi y_\lambda = y_\mu$  for a symmetry  $\sigma = \gamma\pi$  satisfying  $\sigma f = f$  into the same class.

Next, the classes formed will be merged by the second condition of Lemma 4.2.1. Evidently this condition is effective only if there are some  $y_\nu \neq 0$  and  $\sigma$  such that  $\sigma(f \oplus y_\nu^\delta) = f$ . We call such a  $y_\nu^\delta$  as a *conservative linear function* for

$f$  and such a  $\sigma$  as a *restoring symmetry* for  $f$  and  $y_\nu^\varepsilon$ . Here it is observed that any class is composed exclusively of either non-conservative linear functions or conservative linear functions primed or unprimed. Now a little investigation shows that we may proceed as follows. That is: Choose an arbitrary  $y_\nu^\varepsilon$  form a class of conservative linear functions and an arbitrary restoring symmetry  $\sigma$  for  $y_\nu^\varepsilon$ . Merge any two classes if there are a  $y_\lambda$  in one and a  $y_\mu$  in the other such that  $\pi(y_\lambda \oplus y_\nu) = y_\mu$ , where  $\pi$  is the permutation part of  $\sigma$ . Repeat the same procedure for each class of conservative linear functions.

After the classification of odd linear functions is finished, we may choose an arbitrary odd linear function  $y_\lambda$  from each class and form  $f \oplus y_\lambda$ . In this way, we shall obtain the set of representatives of all the genera.

#### *Stage 4: Classification into Types*

In this stage, each genus obtained in Stage 3 will be classified into types. This can be done immediately by complementing the representative of each genus. But on account of its triviality, the process may be entirely omitted. The only non-trivial task is to detect self-complementary genera. Fortunately, however, this is feasible in process of Stage 3 by the following lemma.

*Lemma 4.2.2. Let  $f$  be any function and  $y_\lambda$  be any odd linear function, then  $f \oplus y_\lambda$  is self-complementary if and only if either  $\sigma y_\lambda = y^{-\varepsilon}$  for a symmetry  $\sigma$  such that  $\sigma f = f^\varepsilon$  or  $\sigma(y_\lambda \oplus y_\nu^\varepsilon) = y_\lambda'$  for a symmetry  $\sigma$  and a linear function  $y_\nu^\varepsilon$  such that  $\sigma(f \oplus y_\nu^\varepsilon) = f$ .*

*Proof.* The proof is similar to that of Lemma 4.2.1.

In testing the self-complementarity of a function  $f \oplus y_\lambda$  by Lemma 4.2.2, the following remarks may be of use. That is: In the first place, only functions  $f \oplus y_\lambda$  such that  $f(y_\lambda) = 0$  may be tested. In the second place, we may neglect the possibility  $\sigma f = f'$  by the same reason as stated concerning the first condition of Lemma 4.2. Accordingly,  $f \oplus y_\lambda$  is self-complementary if  $\sigma y_\lambda = y_\lambda'$  for a symmetry  $\sigma$  such that  $\sigma f = f$ . In the third place, it is necessary to consider all the restoring symmetries  $\sigma$  for each conservative linear function  $y_\nu^\varepsilon$  when we use the second condition. Clearly the set of all the restoring symmetries is given by the right coset  $G(f)\sigma$  of the symmetry group  $G(f)$ , where  $\sigma$  is an arbitrary restoring symmetry.

### **4.3. Classification of Boolean Functions of Three Variables**

The classification of Boolean functions of three variables is a well-established problem and there is almost nothing to be added newly by its reinvestigation. The intention here is to apply the method of Section 4.2 to the problem for the purpose of demonstrating its power.

Now we proceed to the classification. Since the sum of squares of eight coordinates of any function of three variables is 16 (Theorem 3.3.1), its index cannot be other than 4, 3 or 2. We shall treat these three cases separately.

#### *4.3.1. Classification of Functions with the Index 4*

Evidently there is only one function with 4 as the first coordinate. It is the function 0 with the coordinates

$$4 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0.$$

Therefore there is a unique prototype and at the same time a unique family of functions with the index 4. Here it is intuitively evident that there are four genera of them represented, for example, by the functions shown in Table 4.3.1. In this table and also in the similar ones appearing hereafter, the numbers standing to the left of the coordinates are the serial numbers of the genera. From Table 4.3.1, we see that the genera "9", "10" and "11" are self-complementary, since they are represented by the linear functions  $x_1$ ,  $x_1 \oplus x_2$  and  $x_1 \oplus x_2 \oplus x_3$  respectively.

Thus, it turns out that functions with the index 4 are classified into a prototype, a family, four genera and five types.

#### 4.3.2. Classification of Functions with the Index 3

Functions having the index 3 as the first coordinate are nothing but atoms, because the first coordinate means that they are of the dimension 1. Therefore there exists a unique prototype and at the same time a unique family of functions with the index 3. As the representative of this prototype (family), we take the atom  $x'_1 x'_2 x'_3$  with the coordinates

$$3 \ \bar{1} \ \bar{1} \ \bar{1} \ \bar{1} \ \bar{1} \ \bar{1} \ \bar{1}.$$

TABLE 4.3.2

"2"	3	$\bar{1}$	$\bar{1}$	$\bar{1}$	$\bar{1}$	$\bar{1}$	$\bar{1}$
"6"	$\bar{1}$	3	$\bar{1}$	$\bar{1}$	$\bar{1}$	$\bar{1}$	$\bar{1}$
"7"	$\bar{1}$	$\bar{1}$	$\bar{1}$	$\bar{1}$	3	$\bar{1}$	$\bar{1}$
"8"	$\bar{1}$	$\bar{1}$	$\bar{1}$	$\bar{1}$	$\bar{1}$	$\bar{1}$	3

In this case also, it is evident that there are four genera represented, for example, by the functions shown in Table 4.3.2.

Thus, it turns out that functions with the index 3 are classified into a prototype, a family, four genera and eight types.

#### 4.3.3. Classification of Functions with the Index 2

The possible absolute values of coordinates of functions with the index 2 are 2 and 0. Hence they must have four coordinates with the absolute value 2 and four coordinates with the value 0 (Theorem 3.3.1). Now, let  $f$  be a function with the index 2 such that  $f(0)=2$ , then  $f(y_i)=2\varepsilon_i$  for some three odd linear functions  $y_1$ ,  $y_2$  and  $y_3$ , and  $f(y_\lambda)=0$  for all other  $y_\lambda$ . Further, since  $f$  is of the dimension 2, we have  $f \leq y_1^{\varepsilon_1} y_2^{\varepsilon_2}$  (Theorem 3.4.1). But  $y_1^{\varepsilon_1} y_2^{\varepsilon_2}$  is also of the dimension 2 ((3.4.4)). Hence we obtain  $f = y_1^{\varepsilon_1} y_2^{\varepsilon_2}$ , and consequently,  $y_1 \oplus y_2 \oplus y_3 = 0$  and  $\varepsilon_1 \varepsilon_2 \varepsilon_3 = -1$  ((3.4.5)). This means that functions having the index 2 as the first coordinates are products of two linear functions other than 0 and 1. Since any two such functions are homologous, there exists a unique prototype of functions with the index 2. As the representative of this prototype, we take the function  $x'_1 x'_2$  whose coordinates are

$$2 \ \bar{2} \ \bar{2} \ 0 \ \bar{2} \ 0 \ 0 \ 0.$$

Next, functions which are homologous to this function, or in other words, functions of the dimension 2 are to be classified into types. In doing this, we



may only consider functions of the standard form (Section 3.7), *i.e.*, those  $f$  with  $f(0)=2$ ,  $f(y_1)=f(y_2)=f(y_3)=\bar{2}$  for some three odd linear functions  $y_1$ ,  $y_2$  and  $y_3$  such that  $y_1 \oplus y_2 \oplus y_3 = 0$  and  $f(y_\lambda) = 0$  for all others. Such functions as these are uniquely determined by the three coordinates  $f(x_1)$ ,  $f(x_2)$  and  $f(x_3)$  by virtue of the relations

$$f(0) + f(x_i) + f(x_j) + f(x_i \oplus x_j) \equiv 0 \pmod{4}$$

and

$$f(x_1) + f(x_2) + f(x_3) + f(x_1 \oplus x_2 \oplus x_3) \equiv 0 \pmod{4}$$

(Theorem 3.4.3).

Thus, it will be easily found that there are three types of functions of the dimension 2 represented, for example, by the functions shown in Table 4.3.3.

Now we shall examine whether the three functions are of different families or not. First, we note that odd linear functions corresponding to the coordinates 2 and  $\bar{2}$  form a group for any of these functions. Then it follows that any even linear function corresponding to the coordinate  $\bar{2}$  is conservative. Hence these three functions are of different families.

TABLE 4.3.3

"3"	2	$\bar{2}$	$\bar{2}$	0	$\bar{2}$	0	0	0
"4"	2	$\bar{2}$	0	0	0	0	$\bar{2}$	$\bar{2}$
"5"	2	0	0	0	$\bar{2}$	$\bar{2}$	$\bar{2}$	0

The three families obtained above are now to be classified into genera. The process and the results of the classification are shown in Table 4.3.4.

	0 1 2 3 1 1 2 1 2 3 3 3 2 3	Symmetry	C.L.	R.S.
"3"	2 $\bar{2}$ $\bar{2}$ 0 $\bar{2}$ 0 0 0 14	(12), [3]	$x_1'$ $x_2'$ $(x_1 \oplus x_2)'$	[2] [1] [12]
"4"	2 $\bar{2}$ 0 0 0 0 $\bar{2}$ $\bar{2}$ 13	(23), [23]	$x_1'$ $(x_2 \oplus x_3)'$ $(x_1 \oplus x_2 \oplus x_3)'$	[2] [1] [12]
"5"	2 0 0 0 $\bar{2}$ $\bar{2}$ $\bar{2}$ 0 12	(1, 2, 3), [123]	$(x_1 \oplus x_2)'$ $(x_1 \oplus x_3)'$ $(x_2 \oplus x_3)'$	[3] [2] [1]
"12"	0 $\bar{2}$ $\bar{2}$ $\bar{2}$ 0 0 0 2			
"13"	0 0 2 $\bar{2}$ $\bar{2}$ 2 0 0			
"14"	0 0 0 2 0 $\bar{2}$ $\bar{2}$ 2			

TABLE 4.3.4

The exact meaning of Table 4.3.4 will now be explained. Each set of symbols in the column "Symmetry" gives a generating system of the symmetry group of the function in the same row. The symbol (1, 2, 3) in the row "5" indicates that the function is symmetric. We adopt this symbol instead of the symbols (12) and (13), because the former is more concise than the latter. Arcs placed under some coordinates show the results of classification of odd linear functions by the first condition of Lemma 4.2.1. Thus, for example, odd linear functions are classified into classes  $[0]$ ,  $[x_1]$ ,  $[x_2]$ ,  $[x_3]$ ,  $[x_1 \oplus x_2]$ ,  $[x_1 \oplus x_3]$ ,  $[x_2 \oplus x_3]$  and  $[x_1 \oplus x_2 \oplus x_3]$

for the function "3", because (12)  $x_1 = x_2$  and (12)  $(x_1 \oplus x_3) = x_2 \oplus x_3$ . Small circles put under some coordinates indicate that the corresponding linear functions, odd or even as the case may be, are conservative. All conservative linear functions are given in the column "C.L." and corresponding typical restoring symmetries in the column "R.S.". Arcs placed over some coordinates indicate mergers of classes of odd linear functions by the second condition of Lemma 4.2.1. Thus, for example, two arcs of the function "3" indicate that the classes  $[x_3]$ ,  $[x_1 \oplus x_3]$ ,  $[x_2 \oplus x_3]$  and  $[x_1 \oplus x_2 \oplus x_3]$  are merged. Arcs indicating mergers of classes of conservative linear functions are omitted, since it is evident that all the classes of conservative linear functions should be merged together. Now it is seen that odd linear functions are classified into two classes for any of the three representatives. Hence any of the three families is classified into two genera of which one is represented by the representative of the family itself and the other is named by the number standing under the arrowhead. A representative of the latter genus can be obtained by the linear addition of an arbitrary odd linear function chosen from the class where the arrow starts. Thus, for example, the function "13" is obtained as the ring sum of  $x_2$  and the function "4". The underlines drawn under the numbers of new genera indicate that they are self-complementary. For example, the genus "13" is self-complementary, because  $[23] x_2 = x_2'$  (The first condition of Lemma 4.2.2).

Thus, it turns out that functions with the index 2 are classified into a prototype, three families, six genera and nine types, since three genera are self-complementary.

#### 4.3.4. Conclusion

The results obtained are tabulated in Table 4.3.5 and Table 4.3.6. In Table 4.3.5, various numbers characterizing the classification of Boolean functions of three variables are collected from the preceding sections. Only the numbers of functions in the last column are added newly. They are obtained by summing up the appropriate numbers given in the column "T" of Table 4.3.6.

TABLE 4.3.5

Index	Prototype	Family	Genus	Type	Function
4	1	1	4	5	16
3	1	1	4	8	128
2	1	3	6	9	112
Total	3	5	14	22	256

In Table 4.3.6, the 14 genera are listed with some useful informations added. The numbers in the column "N" are the serial numbers of genera. The next eight columns give the coordinates of the representatives of genera. The representatives are so chosen or so transformed that their first four coordinates may be positive and the three in the second set may form decreasing sequences. The column "STANDARD SUM" contains the standard sums of the representatives. The column "SYMMETRY" gives generating systems of symmetry groups. The last column "T" contains the numbers of functions which are congruent with the

representatives. They are obtained by dividing the order of the group  $0_n$ , 48 by the orders of symmetry groups.

Surveying the table, we see that the five genera "1", "2", "5", "8" and "11" are symmetric and the three genera "9", "10" and "11" are selfdual. Furthermore, it is noted that any function has a non-trivial symmetry and any function having 0 as the first coordinate is self-complementary. These properties are characteristic of Boolean functions of less than four variables, since they are no longer true for functions of four or more than four variables.

TABLE 4.3.6. Table of Boolean Functions of Three Variables

N					1				Standard sum	Symmetry	T
	0	1	2	3	2	1	1	2			
1	4	0	0	0	0	0	0	0		(1, 2, 3), [1], [2], [3]	1
2	3	1	1	1	$\bar{1}$	$\bar{1}$	$\bar{1}$	1	7	(1, 2, 3)	8
3	2	2	2	0	$\bar{2}$	0	0	0	6 7	(12), [3]	12
4	2	2	0	0	0	0	2	$\bar{2}$	4 7	(23), [23]	12
5	2	0	0	0	$\bar{2}$	$\bar{2}$	$\bar{2}$	0	0 7	(1, 2, 3), [123]	4
6	1	3	1	1	$\bar{1}$	$\bar{1}$	1	$\bar{1}$	5 6 7	(23)	24
7	1	1	1	1	3	1	1	1	1 6 7	(12)	24
8	1	1	1	1	1	1	1	3	3 5 6	(1, 2, 3)	8
9	0	4	0	0	0	0	0	0	4 5 6 7	(23), [2], [3]	6
10	0	0	0	0	4	0	0	0	2 3 4 5	(12), [12], [3]	6
11	0	0	0	0	0	0	0	4	1 2 4 7	(1, 2, 3), [12], [13]	2
12	0	2	2	2	0	0	0	$\bar{2}$	3 5 6 7	(1, 2, 3)	8
13	0	2	2	0	0	2	$\bar{2}$	0	3 4 6 7	(12)[3]	24
14	0	2	0	0	$\bar{2}$	$\bar{2}$	0	$\bar{2}$	0 5 6 7	(23)	24

#### 4.4. Classification of Boolean Functions of Four Variables

Boolean functions of four variables have 16 coordinates and the sum of their squares is equal to 64 (Theorem 3.3.1). Hence the possible values of the index are 8, 7, 6, 5, 4, 3 and 2. These seven cases will be treated successively in the above listed order.

##### 4.4.1. Classification of Functions with the Index 8

Evidently there is only one function with the index 8 as the first coordinate. That is the function 0 whose coordinates are

$$8 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0.$$

Therefore there exists a unique prototype which we call the prototype "8-0" and at the same time a unique family of functions with the index 8. The process and the results of classification of the family are given in the following table which is to be interpreted in the same way as Table 4.3.4.

The representatives of the genera "165", "166", "167" and "168" are not given here for the sake of economy of space. They are given in the table of Boolean

"1"	8	0 0 0 0	0 0 0 0 0 0	0 0 0 0	0	(1, 2, 3, 4), [1], [2], [3], [4]
		↓	↓	↓	↓	
		165	166	167	168	

TABLE 4.4.1

functions of four variables of Appendix 1 together with all others.

Thus, it turns out that functions of the index 8 are classified into a prototype, a family, five genera and six types, because four genera are self-complementary.

#### 4.4.2. Classification of Functions with the Index 7

Functions having the index 7 as the first coordinates are atoms. Therefore there exists a unique prototype which we call the prototype "7-1" of functions with the index 7. As the representative of this family, we take the atom  $x'_1 x'_2 x'_3 x'_4$  whose coordinates are

$$7 \quad \bar{1} \bar{1} \bar{1} \bar{1} \quad \bar{1} \bar{1} \bar{1} \bar{1} \bar{1} \bar{1} \quad \bar{1} \bar{1} \bar{1} \bar{1} \quad \bar{1}.$$

Since atoms are of the same type, the prototype "7-1" consists of a unique family. The process and the results of classification of the family are given in Table 4.4.2.

"2"	7	$\bar{1} \bar{1} \bar{1} \bar{1}$	$\bar{1} \bar{1} \bar{1} \bar{1} \bar{1} \bar{1}$	$\bar{1} \bar{1} \bar{1} \bar{1}$	$\bar{1}$	(1, 2, 3, 4)
		↓	↓	↓	↓	
		109	110	111	112	

TABLE 4.4.2

Thus, it turns out that functions with the index 7 are classified into a prototype, a family, five genera and ten types.

#### 4.4.3. Classification of Functions with the Index 6

Functions with the index 6 have one coordinate with the absolute value 6, seven coordinates with the absolute value 2 and eight coordinates with the value 0 (Theorem 3.3.1, Theorem 3.4.4). Now, let  $f$  be a function such that  $f(0) = 6$ , then, since  $f$  is of the dimension 2, seven odd linear functions corresponding to coordinates with the absolute value 2 together with the function 0 form a group of the dimension 3 (Theorem 3.4.1). Let a base of the group be  $[y_1, y_2, y_3]$  and assume  $f(y_i) = 2 \varepsilon_i$  ( $i = 1, 2, 3$ ). Clearly  $f \leq y_1^{\varepsilon_1} y_2^{\varepsilon_2} y_3^{\varepsilon_3}$ . But  $y_1^{\varepsilon_1} y_2^{\varepsilon_2} y_3^{\varepsilon_3}$  is also of the dimension 2 (Theorem 3.4.2). Therefore we have  $f = y_1^{\varepsilon_1} y_2^{\varepsilon_2} y_3^{\varepsilon_3}$ . It follows then that functions having the index 6 as the first coordinates are products of three independent linear functions. Since any two such functions are homologous, there exists a unique prototype which we call the prototype "6-2-0" of functions with the index 6. As the representative of the prototype, we take the function  $x'_1 x'_2 x'_3$  whose coordinates are

$$6 \quad \bar{2} \bar{2} \bar{2} 0 \quad \bar{2} \bar{2} 0 \bar{2} 0 0 \quad \bar{2} 0 0 0 \quad 0.$$

Next, functions which are homologous to this function, or in other words, functions of the dimension 2 are to be classified into types. In order to do this, we may only consider functions of the standard form, *i.e.*, those for which seven

coordinates with the absolute value 2 are all  $\bar{2}$ . Here it is noted that these functions are completely determined by the four coordinates  $f(x_i)$  ( $i=1, 2, 3, 4$ ) by means of the relations

$$f(0) + f(x_i) + f(x_j) + f(x_i \oplus x_j) \equiv 0 \pmod{4},$$

$$f(x_i) + f(x_j) + f(x_k) + f(x_i \oplus x_j \oplus x_k) \equiv 0 \pmod{4},$$

and 
$$f(0) + f(x_1 \oplus x_2) + f(x_3 \oplus x_4) + f(x_1 \oplus x_2 \oplus x_3 \oplus x_4) \equiv 0 \pmod{4}$$

(Theorem 3.4.3).

Hence they are classified into four types represented by functions whose four coordinates in the second set are  $\bar{2}\bar{2}\bar{2}0$ ,  $\bar{2}\bar{2}00$ ,  $\bar{2}000$  and  $0000$ . The four representatives are shown in Table 4.4.3. Further, these functions are of different families, since none of them is analogous to any other. Therefore they may be taken as the representatives of the four families of the prototype "6-2-0".

The four families are now to be classified into genera. The process and the results of classification are shown in Table 4.4.3.

TABLE 4.4.3

	0	1	2	3	4	1	1	1	2	1	2	2	3	3	3	4	Symmetry
"3"	6	$\bar{2}$	$\bar{2}$	$\bar{2}$	0	$\bar{2}$	$\bar{2}$	0	$\bar{2}$	0	0	$\bar{2}$	0	0	0	0	(1, 2, 3), [4]
		↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	
		59				62			63			180			181		
			172				179										
"4"	6	$\bar{2}$	$\bar{2}$	0	0	$\bar{2}$	0	0	0	0	$\bar{2}$	0	0	$\bar{2}$	$\bar{2}$	$\bar{2}$	(12), (34), [34]
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	
	60				64			65			66			67			
		171				177			178								
"5"	6	$\bar{2}$	0	0	0	0	0	$\bar{2}$	$\bar{2}$	$\bar{2}$	$\bar{2}$	$\bar{2}$	0	0	0	0	(2, 3, 4), [234]
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	
	61							68			69			175		176	
		170				174											
"6"	6	0	0	0	0	$\bar{2}$	$\bar{2}$	$\bar{2}$	$\bar{2}$	$\bar{2}$	$\bar{2}$	$\bar{2}$	0	0	0	0	(1, 2, 3, 4), [1234]
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	
						70											
		169										173				71	

Thus, it turns out that functions with the index 6 are classified into a prototype, four families, 30 genera and 47 types, because 13 genera are self-complementary.

#### 4.4.4. Classification of Functions with the Index 5

Functions with the index 5 have one coordinate with the absolute value 5, three coordinates with the absolute value 3 and 12 coordinates with the absolute value 1 (Theorem 3.3.1, Theorem 3.4.4). Now, let  $f$  be a function such that  $f(0)=5$ . Since  $f$  is of the dimension 3, three odd linear functions corresponding



the six families of the prototype "5-3-1". The process and the results of classification of the six families are also given in Table 4.4.4.

Thus, it turns out that functions with the index 5 are classified into a prototype, six families, 52 genera and 104 types.

#### 4.4.5. Classification of Functions with the Index 4

Functions with the index 4 are classified into two kinds according to whether all coordinates are congruent modulo 4 or not (Theorem 3.4.5).

First, we consider functions of the first kind, those all the coordinates of which are congruent modulo 4. Clearly such functions have four coordinates with the absolute value 4 and 12 coordinates with the value 0 (Theorem 3.3.1). Now, let  $f$  be a function of the first kind such that  $f(0)=4$ . Since  $f$  is of the dimension 4, four odd linear functions corresponding to coordinates with the absolute value 4 form a group of the dimension 2 (Theorem 3.4.1). Transforming  $f$  into the standard form, or rather assuming that  $f$  is already of the standard form from the beginning, we have  $f(0)=4$ ,  $f(y_i)=\bar{4}$  ( $i=1, 2, 3$ ) for some three odd linear functions such that  $y_1 \oplus y_2 \oplus y_3 = 0$  and  $f(y_\lambda)=0$  for all other  $y_\lambda$ . This means that  $f$  is a product of two even linear functions other than 1. Since any two such functions are homologous, we conclude that there exists a unique prototype which we call the prototype "4-0" of functions of the first kind. As the representative of the prototype, we take the function  $x'_1 x'_2$  whose coordinates are

$$4 \quad \bar{4} \quad \bar{4} \quad 0 \quad 0 \quad \bar{4} \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0.$$

Next, functions which are homologous to this function are to be classified into types. But this can be done immediately by analogy with the classification of functions of the dimension 3, because, by replacing 4 with 5,  $\bar{4}$  with  $\bar{3}$  and 0 with 1, the functions under consideration are turned into those considered in Section 4.4.4. Hence we see that there are six types. Their representatives are given below in Table 4.4.5. Further, these six functions are of different families, because, for any of them, any even linear function corresponding to a coordinate  $\bar{4}$  is conservative. Therefore they represent all the six families of the prototype "4-0". The process and the results of classification of these families are given in Table 4.4.5.

The self-complementarity of 13 new genera can be verified by the first condition of Lemma 4.2.2. For example, the genus "183" is self-complementary, because  $[23]x_2 = x'_2$ .

Thus, it turns out that functions of the first kind with the index 4 are classified into a prototype, six families, 19 genera and 25 types.

We now consider functions of the second kind. These functions have two coordinates with the absolute value 4, eight coordinates with the absolute value 2 and six coordinates with the value 0 (Theorem 3.3.1, Theorem 3.4.5). Now, let  $f$  be a function of the second kind such that  $f(0)=4$ . Then, since  $f$  is of the dimension 4 and  $f$  has one more coordinate  $f(y_1) \pm 4$ , it is divisible. Further, let  $x_1^*$ ,  $x_2^*$  and  $x_3^*$  be any three variables such that  $x_1^*$ ,  $x_2^*$ ,  $x_3^*$  and  $y_1$  are mutually independent, and  $L^*$  be the group of odd linear functions determined by the base  $[x_1^*, x_2^*, x_3^*]$ . Then we have  $f(y_\lambda) = \mp f(y_\nu)$  for any odd linear function  $y_\lambda$  of the form  $y_\lambda = y_1 \oplus y_\nu$ , where  $y_\nu \in L^*$  and  $y_\nu \neq 0$  ((3.7.3)). On the other hand, concerning the coordinates  $f(y_\nu)$  for  $y_\nu \in L^*$ , we see that  $[f(y_\nu) - 4\delta_{y_0}]$  ( $y_\nu \in L^*$ )

TABLE 4.4.5

	0	1	2	3	4	1	1	1	2	2	2	3	3	3	3	1	1	1	2	2	3	3	Symmetry	C.L.	R.S.
"13"	4	$\bar{4}$	$\bar{4}$	0	0	$\bar{4}$	0	0	0	0	0	0	0	0	0								(12), (34), [3], [4]	$x'_1$ $x'_2$ $(x_1 \oplus x_2)'$	[2] [1] [12]
"14"	4	$\bar{4}$	0	0	0	0	0	0	$\bar{4}$	0	0	$\bar{4}$	0	0	0								(23), [23], [4]	$x'_1$ $(x_2 \oplus x_3)'$ $(x_1 \oplus x_2 \oplus x_3)'$	[2] [1] [12]
"15"	4	$\bar{4}$	0	0	0	0	0	0	0	0	0	0	0	0	0	$\bar{4}$	$\bar{4}$						(2, 3, 4), [23], [24]	$x'_1$ $(x_2 \oplus x_3 \oplus x_4)'$ $(x_1 \oplus x_2 \oplus x_3 \oplus x_4)'$	[2] [1] [12]
"16"	4	0	0	0	0	$\bar{4}$	$\bar{4}$	0	$\bar{4}$	0	0	0	0	0	0	0	0						(1, 2, 3), [123], [4]	$(x_1 \oplus x_2)'$ $(x_1 \oplus x_3)'$ $(x_2 \oplus x_3)'$	[3] [2] [1]
"17"	4	0	0	0	0	$\bar{4}$	0	0	0	0	$\bar{4}$	0	0	0	0	$\bar{4}$							(12), (34), (13) (24), [12], [34]	$(x_1 \oplus x_2)'$ $(x_3 \oplus x_4)'$ $(x_1 \oplus x_2 \oplus x_3 \oplus x_4)'$	[3] [1] [13]
"18"	4	0	0	0	0	$\bar{4}$	0	0	0	0	0	0	0	0	0	$\bar{4}$	$\bar{4}$	0					(12), (34), [123], [34]	$(x_1 \oplus x_2)'$ $(x_1 \oplus x_3 \oplus x_4)'$ $(x_2 \oplus x_3 \oplus x_1)'$	[3] [2] [1]

represents an indivisible function of the three variables  $x_1^*$ ,  $x_2^*$  and  $x_3^*$  of the dimension 4 ((3.7.4)). Here an inspection of Table 4.3.6 tells us that there are three types of such functions and they are all homologous ("12", "13" and "14" of Table 4.3.6). Furthermore, since it is readily seen that any two functions like  $f$  are homologous, we conclude that there exists a unique prototype which we call the prototype "4-2-0" of functions of the second kind. As the representative of the prototype, we take the function  $\sum(11, 13, 14, 15)$  whose coordinates are

$$4 \ 4 \ 2 \ 2 \ 2 \ \bar{2} \ \bar{2} \ \bar{2} \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ \bar{2} \ 2.$$

Next, functions which are homologous to this function are to be classified into types. In doing this, we may only consider functions  $f$  such that  $f(y_1)=4$  for  $y_1=x_1$ ,  $x_1 \oplus x_2$ ,  $x_1 \oplus x_2 \oplus x_3$ ,  $x_1 \oplus x_2 \oplus x_3 \oplus x_4$ . Then, it is seen that we may put  $x_1^*=x_2$ ,  $x_2^*=x_3$ , and  $x_3^*=x_4$ . On the other hand, concerning the coordinates  $f(y_v)$  for  $y_v \in L^*$ , we may assume  $f(y_2)=f(y_3)=f(y_4)=2$  and  $f(y_2 \oplus y_3 \oplus y_4)=\bar{2}$  for some three independent odd linear functions  $y_2$ ,  $y_3$  and  $y_4$ , and  $f(y_v)=0$  for all other  $y_v$ . In this way, it will be found after a little examination that there are 13 types of functions under consideration and they are of different families. The representatives of the 13 types which are at the same time the representatives of the 13 families of the prototype "4-2-0", are so chosen that the four coordinates



in the second set may form positive decreasing sequences. They are given below in Table 4.4.6 together with the process and the results of classification of families represented by them.

In Table 4.4.6, we see that, among 28 genera of functions of the dimension 8, 12 genera are self-complementary and the other 16 genera are not. The self-

TABLE 4.4.6

	0	1	2	3	4	1	1	1	2	2	3	3	1	1	1	2	2	3	3	Symmetry	C.L.	R.S.
"19"	4	4	2	2	2	2	2	2	0	0	0	0	0	0	0	2	2			(2, 3, 4)	$x_1$	[234]
			82						211							103						
"20"	4	4	2	2	0	2	2	0	2	2	0	2	2	0	0	2	2	0	0	(23) [4]	$x_1$	[23]
			78						215			97				220						
"21"	4	4	2	0	0	2	0	0	2	2	0	2	2	0	2	2	0	2	2	(34)	$x_1$	[2]
			75						89			222				93						
"22"	4	2	2	2	2	2	2	2	0	0	0	0	0	0	2	2	2	2	0	(12), (34)	$(x_1 \oplus x_2)'$	[34]
			74						207			209				104						
"23"	4	2	2	2	2	0	0	0	0	0	0	0	2	2	2	2	2	4	0	(1, 2, 3, 4)	$x_1 \oplus x_2 \oplus x_3 \oplus x_4$	[1234]
			87						208													
"24"	4	2	2	2	0	2	2	0	2	2	2	2	2	0	0	0	2			(12)	$(x_1 \oplus x_2)'$	[123]
			73						198			100				102						
"25"	4	2	2	2	0	0	0	2	0	2	2	2	2	2	2	4	0	0	2	(12)	$x_1 \oplus x_2 \oplus x_4$	[123]
			83						197			212				101						
"26"	4	2	2	0	0	2	2	2	2	2	0	0	0	2	2	0				(12), (34)	$(x_1 \oplus x_2)'$	[12]
			72						201			94				218						
"27"	4	2	2	0	0	0	2	2	2	2	2	4	0	0	2	2	0			(12) [34], (34)	$(x_3 \oplus x_4)'$	[12]
			80						195			217				95						
																219						



TABLE 4.4.7

	0	1	2	3	4	1	1	1	2	2	1	1	1	2	2	Symmetry	C.L.	R.S.
"45"	3	3	3	3	3	1	1	1	1	1	1	1	1	1	3	(1, 2, 3, 4)		
		↓					↓				↓				↓			
		51					152				159				58			
"46"	3	3	3	3	1	1	1	3	1	1	1	1	1	3	1	(23)		
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓			
	48	52			149	151		55	148		162			57				
							154				158				161			
"47"	3	3	3	1	1	1	3	1	1	3	3	1	1	1	1	(12)(34)		
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓			
	49				147	150		54	156		56			160	155	163		
"48"	3	3	3	1	1	1	3	3	1	1	1	1	1	1	3			
"49"	3	3	3	1	1	1	3	1	1	1	1	1	1	1	1			
"50"	3	3	3	1	1	1	1	1	1	1	1	1	1	3	3	(12), (34)	$x_1 \oplus x_2 \oplus x_3 \oplus x_4$	(13)(24)
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓			
	53				157	146					153			164				
"51"	3	3	1	1	1	3	3	3	1	1	1	1	1	3	1			
"52"	3	3	1	1	1	3	3	1	1	3	1	1	1	3	1			
"53"	3	3	1	1	1	3	1	1	3	3	1	1	1	3	1			
"54"	3	3	1	1	1	3	1	1	3	1	1	1	3	1	3			
"55"	3	3	1	1	1	3	1	1	1	1	1	3	3	1	3			
"56"	3	1	1	1	1	3	3	1	1	3	1	3	3	1	1			
"57"	3	1	1	1	1	3	3	1	1	1	1	3	1	1	3			
"58"	3	1	1	1	1	1	1	1	1	1	1	3	3	3	3			

Thus, it turns out that functions with the index 3 are classified into a prototype, four families, 33 genera and 66 types.

#### 4.4.7. Classification of Functions with the Index 2

Clearly every coordinate of functions with the index 2 has the absolute value 2 (Theorem 3.3.1). Let  $f$  be a function with the index 2 such that  $f(0)=2$ . Without loss of generality, we may assume  $f \geq x'_1 x'_2 x'_3 x'_4$ , or equivalently,  $\sum_L f(y_\lambda) = -8$  (Theorem 3.4.6), since any function other than 0 can be transformed into such a function, if necessary, by a suitable reflexion operator. Under this assumption,  $f$  has six coordinates with the value 2 and ten coordinates with the value  $\bar{2}$ . Now, let  $y_i$  be the five odd linear functions such that  $y_i \neq 0$  and  $f(y_i)=2$  ( $i=1, 2, 3, 4, 5$ ). When we choose any four, say,  $y_1, y_2, y_3$  and  $y_4$  out of the five, at least three of them, say,  $y_1, y_2$  and  $y_3$  are independent. Let the group of odd linear functions determined by the base  $[y_1, y_2, y_3]$  be denoted by  $L$ . If we assume that at least one of  $y_4$  and  $y_5$  belongs to  $L$ , we obtain  $\sum_L f(y_\lambda)=4$  or  $\sum_L f(y_\lambda)=8$ . But these are both impossible, because  $\sum_L f(y_\lambda)$  must be equal to 0 or  $-8$  when  $f \geq x'_1 x'_2 x'_3 x'_4$  (Theorem 3.4.7). It follows that any four out of the five  $y_i$  must be independent. Since there is no set of five independent odd linear functions, we conclude that  $y_1 \oplus y_2 \oplus y_3 \oplus y_4 \oplus y_5 = 0$ . Therefore any two functions like  $f$  are homologous. Consequently, there exists a unique prototype which we call the prototype "2-2" of functions with the index 2. As its representative, we take the function  $\sum(0, 7, 11, 13, 14, 15)$  whose coordinates are

$$2 \ 2 \ 2 \ 2 \ 2 \ \bar{2} \ \bar{2} \ \bar{2} \ \bar{2} \ \bar{2} \ \bar{2} \ 2 \ \bar{2} \ \bar{2} \ \bar{2} \ 2.$$

Next, functions which are homologous to this function, *i.e.*, those with the index 2 as the first coordinate are to be classified into types. Here it is noted that functions with the index 2 as the first coordinate can be turned into those with the index 3 as the first coordinate by replacing 2 with 3 and  $\bar{2}$  with  $\bar{1}$ . Therefore, in classifying functions under consideration, we may only take 14 functions corresponding to the representatives of the 14 families of the prototype "3-1". It is plausible, however, that some of these 14 functions are congruent, because they have further symmetry structures based on the uniformity of the absolute value of coordinates which are not possessed by those of prototype "3-1". In fact, by transforming the 14 functions in such a manner that the four coordinates in the second set may become positive, it will be found that any two of these functions are of the same type and at the same time of the same family, if and only if they correspond to those of the same family of the prototype "3-1". Thus, we conclude that functions with the index 2 as the first coordinate are classified into four types. Their representatives are shown in Table 4.4.8.

TABLE 4.4.8

"105"	2	2	2	2	2	$\bar{2}$	$\bar{2}$	$\bar{2}$	$\bar{2}$	$\bar{2}$	$\bar{2}$	$\bar{2}$	$\bar{2}$	$\bar{2}$	2
"106"	2	2	2	2	2	$\bar{2}$	$\bar{2}$	$\bar{2}$	$\bar{2}$	$\bar{2}$	2	2	2	$\bar{2}$	2
"107"	2	2	2	2	2	$\bar{2}$	$\bar{2}$	2	2	$\bar{2}$	2	2	$\bar{2}$	$\bar{2}$	$\bar{2}$
"108"	2	2	2	2	2	$\bar{2}$	2	2	2	2	$\bar{2}$	$\bar{2}$	$\bar{2}$	$\bar{2}$	2

The above four functions represent all the four genera of the prototype "2-2", since any function which is analogous to any of them has the first coordinate with the absolute value 2. Incidentally it may be added that the four functions constitute a counter example against the possibility of generalization of Theorem 3.5.2 and Theorem 3.5.4 to functions of even dimensions.

Thus, it turns out that functions with the index 2 are classified into a prototype, four families, four genera and eight types.

#### 4.4.8. Conclusion

The results of the classification of Boolean functions of four variables are tabulated in Table 4.4.9 and Table of Boolean Functions of Four Variables in Appendix 1.

In Table 4.4.9, various values characterizing the classification are collected from the preceding sections. Only the numbers of functions in the last column are added newly. They are obtained by summing up the numbers given in the column "T" of Table of Boolean Functions appropriately.

TABLE 4.4.9

Index	Prototype	Family	Genus	Type	Function
8	1("8-0")	1	5	6	32
7	1("7-1")	1	5	10	512
6	1("6-2-0")	4	30	47	3,840
5	1("5-3-1")	6	52	104	17,920
4	1("4-0")	6	19	25	1,120
4	1("4-2-0")	13	74	136	26,880
3	1("3-1")	4	33	66	14,336
2	1("2-2")	4	4	8	896
Total	8	39	222	402	65,536

Thus, 65, 536 Boolean functions of four variables are classified into 8 prototypes, 39 families, 222 genera and 402 types.

Muller studied the coordinates of Boolean functions of four variables by a digital computer. He calculated the numbers of coordinates of the absolute values 0, 1, 2, . . . , 8 for each function, and put all the functions for which these nine numbers agreed into the same equivalence class. In this way, he has found that there are 8 such equivalence classes. Obviously the 8 classes are nothing but the 8 prototypes in the present terminology. But it is doubtful that Muller's equivalence classes and prototypes are identical for functions of more than four variables.

In Table of Boolean Functions of Four Variables in Appendix 1, the 222 genera are listed with some useful informations added. The first column "N" gives the serial numbers of genera. The next 16 columns give the coordinates of representatives of genera. They are so chosen that their first five coordinates may become non-negative and the four in the second set may form decreasing sequences. The column "Standard Sum" gives the standard sums of the representatives. The column "Symmetry" contains generating systems of symmetry

groups of the representatives. When the column contains an entry “-”, the corresponding representative is perfectly asymmetric. The column “T” gives the numbers of functions which are congruent with the representatives. They are obtainable by dividing the order of the group  $O_i$ , 384 by the orders of symmetry groups. The last column “Remarks” contains some symbols specifying certain particular structures of the representatives or genera. The meanings of the symbols appearing in the column are as follows. The digits 0, 1, 2 and 3 give the numbers of essential variables of degenerate representatives. The symbols  $S_{a,b,\dots,k}$  means that the corresponding representative is the symmetric function with the  $a$ -numbers  $a, b, \dots, k$ . The symbols FS, SD and M are the abbreviations of “functionally separable”, “self-dual” and “monotonous” respectively. Among the structures given by the column, those specified by the symbols  $S_{a,b,\dots,k}$  and  $M$  are not the properties of genera. However, it will be observed from the way of choice of the representatives that, when a genus contains a symmetric (monotonous) function, its representative is symmetric (monotonous).

There are 58 genera of functions of the dimension 8 (having 0 as the first coordinate) of which 42 are self-complementary and the other 16 belonging to the prototype “4-2-0” are not (see Section 4.4.5). These 16 genera are distinguished from self-complementary ones by splitting each of them into two complementary types. Thus, for example, the genus “197” is split into two types “197 a” and “197 b”.

We now explain how to identify the genus or the type of a given function by the table.

*Step 1:* Expand the given function  $f$  into the standard sum.

*Step 2:* Examine the dimension of  $f$ . If it is greater than 8, take  $f'$  instead of  $f$ . Calculate the coordinates of  $\bar{f}$  by Table 3.3.2 or the same table relisted in Appendix 1, where  $\bar{f}$  is  $f$  or  $f'$  as the case may be.

*Step 3:* Examine the four coordinates in the second set, and apply a symmetry  $\sigma$  to  $\bar{f}$ , if necessary, to turn the four coordinates into a non-negative decreasing sequence.

*Step 4:* Look for the function  $\sigma\bar{f}$  in the table. If it appears there, the process terminates. But if it does not appear there,  $\sigma\bar{f}$  must be transformed somehow. Clearly, in this case, we may only consider such symmetries  $\tau$  as preserve the four coordinates of  $\sigma\bar{f}$  as a whole. Testing every possible symmetry with the above property, we eventually arrive at a function  $\tau\sigma\bar{f}$  appearing in the table. Under most circumstances, however, the choice of a correct symmetry will be suggested by the inspection of other coordinates. Sometimes it is even possible to identify the genus or the type without transforming  $\bar{f}$  further.

The rules stated above will now be illustrated by the following examples.

*Example 4.4.1.* Identify the genus (type) of the function:

$$f = \sum (1, 2, 3, 7, 11, 12).$$

*Solution.* Calculating the coordinates by Table 3.3.2, we obtain

$$f: \quad 2 \quad \bar{2} \quad \bar{2} \quad 2 \quad 2 \quad \bar{2} \quad 2 \quad 2 \quad 2 \quad 2 \quad \bar{2} \quad \bar{2} \quad \bar{2} \quad 2 \quad 2.$$

In order to remove the negative signs of  $f(x_1)$  and  $f(x_2)$ , we apply the reflexion operator  $[12]$  to  $f$ . Thus, we obtain

$$[12]f: 2 \ 2 \ 2 \ 2 \ \bar{2} \ \bar{2} \ \bar{2} \ \bar{2} \ \bar{2} \ \bar{2} \ \bar{2} \ \bar{2} \ \bar{2} \ \bar{2} \ 2.$$

This function appears as the representative of the genus "105". Hence  $f$  belongs to the genus "105" (the type "105"). Since  $[12]f$  is symmetric,  $f$  is symmetric with respect to  $x'_1, x'_2, x_3$  and  $x_4$ .

*Example 4.4.2.* Identify the genus (type) of the function:

$$f = \sum(2, 3, 4, 6, 8, 10, 11, 12).$$

*Solution.* Calculating the coordinates by Table 3.3.2, we obtain

$$f: 0 \ 0 \ \bar{2} \ 2 \ \bar{4} \ 2 \ 2 \ 0 \ 4 \ 2 \ \bar{2} \ 0 \ 2 \ 2 \ 0 \ 0.$$

First, we apply the reflexion operator  $[24]$  to remove the negative signs of  $f(x_2)$  and  $f(x_4)$  and obtain

$$[24]f: 0 \ 0 \ 2 \ 2 \ 4 \ \bar{2} \ 2 \ 0 \ \bar{4} \ 2 \ 2 \ 0 \ 2 \ \bar{2} \ 0 \ 0.$$

Next, we apply the permutatition operator  $(14)$  to rearrange  $0224$  into  $4220$  and obtain

$$(14)[24]f: 0 \ 4 \ 2 \ 2 \ 0 \ 2 \ 2 \ 0 \ \bar{4} \ \bar{2} \ 2 \ 0 \ 2 \ \bar{2} \ 0 \ 0.$$

But this function does not appear in the table. Hence it must be transformed somehow. Clearly, possible symmetries to be applied are  $[4]$ ,  $(23)$  and  $(23)[4]$ , because only these preserve the four coordinates  $4220$  as a whole. Thus, applying  $[4]$  tentatively, we obtain

$$[124](14)f: 0 \ 4 \ 2 \ 2 \ 0 \ 2 \ 2 \ 0 \ \bar{4} \ 2 \ \bar{2} \ 0 \ \bar{2} \ 2 \ 0 \ 0,$$

because  $[4](14)[24] = [4][12](14) = [124](14)$ . This function appears as the representative of the type "200 b". Therefore  $f$  belongs to the genus "200" (the type "200 b"). Since  $[124](14)f$  has only one non-trivial symmetry  $(23)[4]$ ,  $f$  has only one non-trivial symmetry  $(23)[123] = [124](14)(23)[4]([124](14))^{-1}$ .

*Example 4.4.3.* Identify the genus (type) of the function:

$$f = \sum(0, 1, 2, 4, 5, 9, 10, 11, 14).$$

*Solution.* Since  $d(f) = 9 > 8$ , we take the complement:

$$f' = \sum(3, 6, 8, 12, 13, 15).$$

Calculating the coordinates by Table 3.3.2, we obtain

$$f': 1 \ 1 \ 3 \ 1 \ 1 \ \bar{1} \ 5 \ 1 \ \bar{1} \ \bar{1} \ \bar{3} \ \bar{1} \ 3 \ 1 \ \bar{1} \ \bar{1}.$$

We apply  $(12)$  to rearrange  $1311$  into  $3111$  and obtain

$$(12)f': 1 \ 3 \ 1 \ 1 \ 1 \ \bar{1} \ \bar{1} \ \bar{1} \ 5 \ 1 \ \bar{3} \ \bar{1} \ 3 \ \bar{1} \ 1 \ \bar{1}.$$

Here it may be observed that, among 11 representatives having the four coordinates 3111 in the second set, only the representative of the genus "130" has the coordinates 5 and  $\bar{3}$  in the third set. Therefore  $f$  must belong to the genus "130" (the type "130"). In fact, transforming  $(12)f'$  by (23), we obtain

$$(123)f': 1 \ 3 \ 1 \ 1 \ 1 \ \bar{1} \ \bar{1} \ \bar{1} \ 5 \ \bar{3} \ 1 \ \bar{1} \ \bar{1} \ 3 \ 1 \ \bar{1},$$

the representative of the genus "130". Since  $(123)f'$  is perfectly asymmetric,  $f$  is so too.

#### 4.4.9. Comparison of the New Table with the Harvard Table

As we mentioned in Section 4.1, the classification of Boolean functions of four variables has been carried out by the Staff of the Computation Laboratory of Harvard University. The results have been published in the Appendix to the book "Synthesis of Electronic Computing and Control Circuits". The Appendix contains three tables, of which Table 1.3 (Harvard Table) is the main table listing the 402 types of Boolean functions of four variables, and the other two are auxiliary tables. The Harvard Table has six columns and from left to right they give:

1. The dimensions.
2. The serial numbers of types.
3. The standard sums of the representatives of types.
4. Standard vacuum tube circuits realizing the representatives.
5. The numbers of control grids necessary in the circuits.
6. The numbers of functions which are congruent with the representatives.

As the representative of each type, the function having the earliest standard sum in the lexicographical order with respect to the atoms  $0, 1, \dots, 15$  is chosen. The representatives are grouped according to their dimensions, and, in each group, they are arranged in the lexicographical order.

The process for the identification of the type of a given function  $f$  is as follows. First,  $f$  will be transformed, if necessary, into a function including the atom 0 by a reflexion operator  $r$ . Table 1.1 is prepared for this purpose. It contains all the results of applications of the 16 reflexion operators to the 16 atoms. Next, the function  $rf$  is to be looked for in Table I.2. It contains about 3,000 functions and, for each of them, the serial number of the type and a permutation operator transforming it into the representative of the type are given. Thus, when the function  $rf$  is found in Table I.2, the process terminates. But, when it is not found there, the process must be repeated from the beginning by choosing another reflexion operator. In this way, the process will be continued until, for a certain reflexion operator  $r^*$ ,  $r^*f$  is found in Table I.2. Since no criterion is available for the choice of the correct reflexion operator, the process may be short or long as the case may be.

We now proceed to the comparison of the New Table with the Harvard Table. The following items may be regarded as the relative merits of the New Table.

- a. The New Table contains the informations concerning the symmetry structures and several other structures of Boolean functions which are not contained in the Harvard Table.



b. In the New Table, it is clarified that 42 types among the 74 types of functions of dimension 8 are self-complementary and the remaining 32 types are not, while, in the Harvard Table, this fact is altogether overlooked.

c. In the New Table, each representative of the type manifests the structure of the type straightforwardly, but, in the Harvard Table, the same is not always the case.<sup>1)</sup>

d. The New Table does not require an extensive auxiliary table like Table I.2 of the Harvard Table.

Fairly speaking, however, only the first two items should be considered as the decisive merits of the New Table. On the other hand, the New Table has following drawbacks.

a. The New Table requires the column for standard sums besides the column for the coordinates in order to express the representatives explicitly.

b. The New Table requires somewhat cumbersome calculations for identifying the type of a given function.

These drawbacks, however, are not considered as serious, because, in the first place, the drawback *b* is compensated for to some extent by the ease of the choice of the correct or at least reasonable reflexion operator. The New Table contains no information corresponding to the columns 4 and 5 of the Harvard Table. But it is not a drawback, because it is irrelevant for the present comparison.

In conclusion, it may be added that the perfect agreement of the two tables has been verified by an exhaustive comparison.

## 5. Synthesis of Relay Switching Circuits

### 5.1. Relay Switching Circuits

In a relay switching circuit, the input is specified by whether each relay is *energized* or *de-energized*, and the output is specified by whether the circuit is closed or open. Let us represent the states of relays with the variables each of which takes the value 0 or 1 according as the corresponding relay is de-energized or energized, and the state of the circuit with the variable which takes the value 0 or 1 according as the circuit is open or closed. The former variables are called the *input variables* and the latter variable is called the *output variable*. Ordinarily, the output is uniquely determined by the input, so the output variable is a binary function, and therefore, a Boolean function<sup>1)</sup> of the input variables. This is the reason why the Boolean algebra is an indispensable tool for the analysis and the synthesis of relay and other switching circuits.

Now, before proceeding further, two remarks will be stated. In the first place, the above mentioned simple functional relation between the input and the output holds only for one kind of switching circuit called *combinational*, but does not for another kind called *sequential*. In a sequential switching circuit, the output is determined not only by the present input but also by the past inputs. Accordingly, the output variable is a Boolean function not only of the present input

<sup>1)</sup> For example, each representative of symmetric types is a symmetric function in the New Table, but the same is not always the case in the Harvard Table.

<sup>2)</sup> See page 8.

variables but also of the past input variables. Nonetheless, the Boolean algebra is still very useful for sequential circuits as well. At any rate, we are concerned only with combinational switching circuits in this paper and we shall omit the adjective "combinational" hereafter.

In the second place, there are two formulations or languages in dealing with switching circuits by Boolean algebras. One is called the *transmission language* and the other the *hindrance language*. In the transmission language, the input and the output variables are defined as above, while, in the hindrance language, they are defined with the roles of 0 and 1 interchanged. The output variable is called the *transmission* in the transmission language, and the *hindrance* in the hindrance language. Thus, the transmission (hindrance) is 1 (0) if the circuit is closed, and is 0 (1) if the circuit is open. These two languages are perfectly dual to one another and there is no essential reason for discriminating between them. But the recent tendency seems to be in favor of the transmission language, and so we adopt it also in this paper.

We now return to the subject and see how relay switching circuits are analyzed by Boolean algebras. To begin with, we shall find the transmission of a single contact. Let  $x$  be the input variable of the relay  $X$ , and  $m(x)$  and  $b(x)$  be the transmissions of a *make contact* and a *break contact* of  $X$  respectively. Then, since every make (break) contact is closed or open (open or closed) according as the relay is energized or de-energized, we obtain the following table which indicates that

$$m(x) = x \quad \text{and} \quad b(x) = x'.$$

TABLE 5.1.1

Relay	$x$	Make Contact $m(x)$		Break Contact $b(x)$	
energized	1	closed	1	open	0
de-energized	0	open	0	closed	1

Thus, it has been shown that the transmission of a make contact is the input variable of the relay itself and the transmission of a break contact is the complement of the input variable of the relay.

Next, we shall find the transmissions of a *parallel connection* and a *series connection* of two circuits. Let  $f$  and  $g$  be the transmissions of two circuits, and  $p$  and  $s$  be the transmissions of their parallel connection and series connection respectively. Then, since the parallel connection is closed if and only if at least one of the component circuits is closed, and the series connection is closed if and only if both of the component circuits are closed, we obtain the following table which indicates that

TABLE 5.1.2

$f$	$g$	$p$	$s$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

$$p = f + g \quad \text{and} \quad s = fg.$$

Thus, it has been shown that the transmission of a parallel (series) connection is the *sum* (*product*) of the transmissions of component circuits.

Now it is easy to analyze a *series-parallel* circuit,

because its transmission will be written down immediately from the circuit diagram by the above rule.

*Example 5.1.1.* Analyze the series-parallel circuit shown in Fig. 5.1.1.

*Solution.* The transmission is written down immediately as:

$$f = xz'y + (w + x')(w' + y').$$

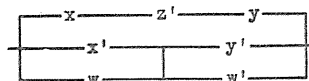


FIG. 5.1.1

The analysis of a *non-series-parallel* circuit, or, a *bridge* is a little complicated but it is still easy. Since a bridge is closed if and only if at least one of the *paths* between terminals is closed, its transmission is given by the sum of transmissions of all paths.

*Example 5.1.2.* Analyze the bridge shown in Fig. 5.1.2.

*Solution.* The circuit has four paths  $xy$ ,  $vw$ ,  $xuw$  and  $vuy$ . So the transmission is given by:

$$f = xy + vw + xuw + vuy.$$

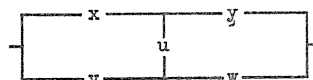


FIG. 5.1.2

Thus, there is no difficulty so far as the analysis is concerned.

As regards the synthesis, on the contrary, the situation is entirely different. Of course, it is always possible to synthesize a circuit, for example, a series-parallel circuit with the given transmission, but that is not enough. What is required is to synthesize a circuit with the given transmission which is as simple as possible under a certain standard of simplicity. There are two common standards of simplicity: One is the number of contacts and the other is the number of springs. At first sight, these two standards seem to be identical, because every contact requires two springs. But the fact is that, by combining any pair of a make contact and a break contact of the same relay sharing a node into a *transfer contact*, we can save one spring per one transfer contact. Thus, occasionally it happens that a circuit which is minimal in contacts is not minimal in springs and vice versa.

The search for a general method for synthesizing a circuit minimal in contacts or in springs is the central problem in the synthesis of relay switching circuits. But, unfortunately, there is no such method available at the present. Indeed, it is very difficult even to determine whether a given circuit may be minimal (in contacts or in springs) or not. Perhaps the only general criterion for the minimality of circuits may be the following: Let  $C$ ,  $S$  and  $E$  be the number of contacts, the number of springs and the number of *essential contacts* respectively, where a contact is said to be essential when its transmission is equal to an essential literal of the transmission function of the circuit. Then, since obviously every essential contact is indispensable for the circuit, we have  $C \geq E$ . Therefore if  $C = E$ , the circuit is minimal in contact. Next, concerning the relation between  $C$  and  $S$ , we have  $S \geq [(3C + 1)/2]$ , because the largest possible number of transfer contacts is  $[C/2]$ . Hence, if  $C$  is minimal and  $S \leq [3C/2] + 2$ , there is no circuit

which is not minimal in contacts but is minimal in springs.

**Example 5.1.3.** Determine whether the circuit shown in Fig. 5.1.3. may be minimal or not.

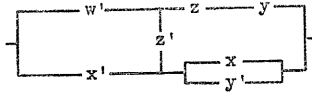


FIG. 5.1.3

**Solution.** The transmission of the circuit is given by:

$$f = x'y' + w'xz' + w'yz.$$

This is the minimal sum and we obtain  $E=7$ .

Further, we obtain  $C=7$  and  $S=11$  from the circuit diagram. Therefore  $C=E$  and  $S=[(3C+1)/2]$ . Accordingly, the circuit is minimal both in contacts and in springs.

Now we are going to explain various methods for synthesizing relay switching circuits in the following sections. The methods may be very effective and lead to minimal circuits sometimes but in a particular problem they may be of little use.

## 5.2. Expansion Method

One of the most general and powerful methods is the *expansion method*. In this method, the transmission function is expanded with respect to one or more variables and then a circuit is synthesized directly on the basis of the result of expansion. This method can be used extensively and, under most circumstances, results in a bridge, planar or non-planar. Furthermore, circuits synthesized by this method have usually many transfer contacts. There are several varieties of the method, of which the *disjunctive tree method*<sup>1)</sup> is fundamental.

### 5.2.1. Disjunctive Tree Method

The disjunctive tree method is based on the interconnection of two networks shown in Fig. 5.2.1. Boxes  $M$  and  $N$  represent networks with  $n+1$  terminals each. The network  $M$  has the transmission  $u_i$  ( $i=1, 2, \dots, n$ ) between terminals  $a$  and  $i$ ; the network  $N$  has the transmission  $v_i$  ( $i=1, 2, \dots, n$ ) between terminals  $b$  and  $i$ ;  $M$  is a *disjunctive* network, i.e., the transmission between any pair of terminals  $i$  and  $j$  is 0.

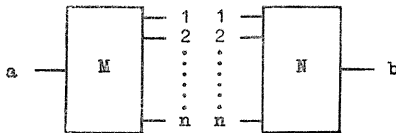


FIG. 5.2.1

With these assumptions, the following theorem was proved by Shannon: "If the corresponding terminals  $1, 2, \dots, n$  of  $M$  and  $N$  are connected together, the transmission  $f$  between the terminals  $a$  and  $b$  is given by  $f = \sum u_i v_i$ ."

In view of the theorem, the disjunctive tree method proceeds as follows:

(1) Write the given transmission  $f$  in the form  $f = \sum u_i v_i$ . Usually the expression will be obtained by expanding  $f$  with respect to some variables. Then,  $u_i$  are products of literals of the expanding variables and  $v_i$  are the correspond-

<sup>1)</sup> cf. Ref. 36.

ing coefficients.

(2) Construct a disjunctive network  $M$  realizing  $u_i$  functions. Usually  $M$  is a disjunctive tree.

(3) Construct an  $N$  network realizing  $v_i$ .

(4) Connect the corresponding terminals of  $M$  and  $N$ . For a terminal of  $N$  network corresponding to  $v_i = 0$ , no connection is needed and the corresponding element in  $M$  network can be eliminated.

Let us now apply the method to transmission functions of three variables. By the expansion theorem, any function of three variables can be written in the form:

$$f(x, y, z) = x'y'f(0, 0, z) + x'yf(0, 1, z) + xy'f(1, 0, z) + xyf(1, 1, z).$$

Putting	$u_1 = x'y',$	$v_1 = f(0, 0, z),$
	$u_2 = x'y,$	$v_2 = f(0, 1, z),$
	$u_3 = xy',$	$v_3 = f(1, 0, z),$
	$u_4 = xy,$	$v_4 = f(1, 1, z),$

we obtain 
$$f(x, y, z) = \sum_{i=1}^4 u_i v_i.$$

The  $M$  network realizing  $u_i$  is the disjunctive tree of Fig. 5.2.2. The  $v_i$  are functions of the variable  $z$ . Therefore they must be selected from the set  $[0, 1, z, z']$ . Evidently, the largest  $N$  network is the one shown in Fig. 5.2.2. Thus, we see that any function of three variables can be realized with not more than 8 contacts.

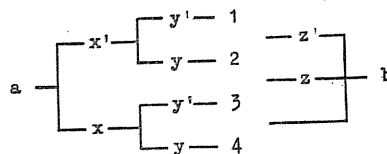


FIG. 5.2.2

Functions of four variables can be realized in either of two ways. In one way, the function will be expanded with respect to three of the variables. Then the  $N$  network is again to realize functions of a single variable, and therefore, the largest  $N$  network is the same as that of Fig. 5.2.2. The  $M$  network is a disjunctive tree containing 14 contacts if all  $u_i$  are present. Accordingly, the combined circuits have at most 16 contacts. In another way, the function will be expanded with respect to two of the variables. When a function  $f(w, x, y, z)$  is expanded with respect to  $w$  and  $x$ , we have

$$f(w, x, y, z) = w'x'f(0, 0, y, z) + w'xf(0, 1, y, z) + wx'f(1, 0, y, z) + wxf(1, 1, y, z).$$

The  $u_i$  and  $v_i$  are now identified as:

$u_1 = w'x',$	$v_1 = f(0, 0, y, z),$
$u_2 = w'x,$	$v_2 = f(0, 1, y, z),$
$u_3 = wx',$	$v_3 = f(1, 0, y, z),$
$u_4 = wx,$	$v_4 = f(1, 1, y, z).$

The  $M$  network is a disjunctive tree similar to that of Fig. 5.2.2. The  $v_i$

are now functions of two variables and must be selected from the set of 16 such functions. Shannon classified all functions of two variables as shown in the following table.

TABLE 5.2.1

A	0, 1.
B	$y, y', z, z'$ .
C	$y' + z', y' + z, y + z', y + z$ .
D	$y'z', y'z, yz', yz$ .
E	$y \oplus z, (y \oplus z)'$ .

By the help of this classification, Shannon studied all the possible selections of functions of two variables exhaustively and gave the upper bound of 14 contacts for the combined circuits.<sup>1)</sup> We shall not discuss how this upper bound was obtained. Here we shall show some typical cases in the following examples.

*Example 5.2.1.* Synthesize a circuit with the transmission:

$$f = w'y' + x'y' + wx'z' + w'xz' + w'x'z + wxyz.$$

*Solution.* Expanding  $f$  with respect to  $w$  and  $x$ , we obtain

$$f = w'x'(y' + z) + w'x(y' + z') + wx'(y' + z') + wxyz.$$

The  $u_i$  and  $v_i$  are identified as:

$$\begin{aligned} u_1 &= w'x', & v_1 &= y' + z, \\ u_2 &= w'x, & v_2 &= y' + z', \\ u_3 &= wx', & v_3 &= y' + z', \\ u_4 &= wx, & v_4 &= yz. \end{aligned}$$

We note that  $v_2 = v_3$ , so the terminals 2 and 3 of the  $M$  network will be connected to a single terminal  $y' + z'$  of the  $N$  network. The best  $N$  network is shown in Fig. 5.2.3. The two functions  $y' + z$  and  $y' + z'$  are realized by the three terminal network containing only three contacts, and the function  $yz$  is obtained by adding only one contact  $y$ . When this  $N$  network is connected to the  $M$  network, the circuit shown in Fig. 5.2.4 will be obtained. The circuit is a non-planar bridge and requires 10 contacts and 15 springs.

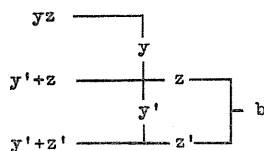


FIG. 5.2.3

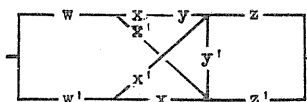


FIG. 5.2.4

<sup>1)</sup> The real upper bound is not 14 but 13. See page 179.

In this example, all the  $v_i$  are selected from the groups  $C$  and  $D$ . Usually, it is easy to treat the cases where all the  $v_i$  are selected from the groups  $A$ ,  $B$ ,  $C$  and  $D$ .

*Example 5.2.2.* Synthesize a circuit with the transmission:

$$f = wx'y + wx'z + wyz + wxy'z' + w'xyz' + w'xy'z.$$

*Solution.* Expanding  $f$  with respect to  $w$  and  $x$ , we obtain

$$f = w'x(y'z + yz') + wx'(y + z) + wx(y'z' + yz).$$

This time,  $v_1 = 0$ , and  $v_2$  and  $v_4$  are from the group  $E$ . The best  $N$  network is given in Fig. 5.2.5. The two functions  $y'z + yz'$  and  $y'z' + yz$  are realized by a network which contains the three-terminal network similar to that used in the preceding example, and is tapped without any additional contact to yield the function  $y + z$ .

The combined circuit is shown in Fig. 5.2.6. It is a planar bridge and requires 12 contacts and 19 springs.

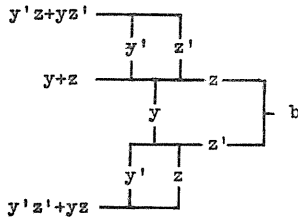


FIG. 5.2.5

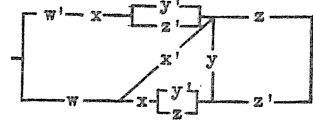


FIG. 5.2.6

Next, we change the variables of expansion to  $y$  and  $z$ . The result of the expansion is

$$f = y'z'wx + y'z(w'x + wx') + yz'(w'x + wx') + yzw.$$

The  $v_i$  are now identified as  $v_1 = wx$ ,  $v_2 = v_2 = w'x + wx'$  and  $v_4 = w$ . The best  $N$  network and the combined circuit are shown in Fig. 5.2.7 and Fig. 5.2.8 respectively. The circuit is non-planar bridge and requires 11 contacts and 17 springs. Therefore it is simpler than the previous circuit by one contact and two springs.

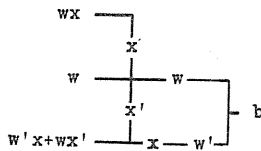


FIG. 5.2.7

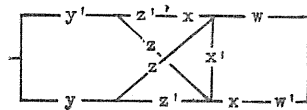


FIG. 5.2.8

As illustrated in the above example, the quality of circuits depends upon the choice of expanding variables so that it is necessary to try every possible choice to obtain the best circuit.

Transmission functions of more than four variables can be treated also by the disjunctive tree method. But, in such cases, there are many ways for distributing variables between the  $M$  and  $N$  networks. For example functions of six variables can be realized with three variables in both the  $M$  and  $N$  networks, with two in the  $M$  network and four in the  $N$  network, or vice versa. Because of the complexity of the problem, almost nothing has been found which is general enough to serve as a guide.

The disjunctive tree method is only one of the varieties of the expansion method and several others are conceivable. Here the author proposes the following three varieties which seem to be new. They are the *sandwich method*, *cross connection method* and *square connection method*. They are based on the expansion of the transmission function with respect to two variables. Although their scopes of application are restricted somehow, they have a common advantage over the disjunctive tree method in that they require only four contacts for the expansion part, while the disjunctive tree method requires six contacts for the  $M$  network.

### 5.2.2. Sandwich Method

Assume that the given transmission function  $f$  is expanded with respect to two variables, say,  $x$  and  $y$  as:

$$f = x'y'v_1 + x'yv_2 + xy'v_3 + xyv_4.$$

The general principle of the sandwich method is to realize this function by the circuit shown in Fig. 5.2.9. The expansion part is split into two parts on both

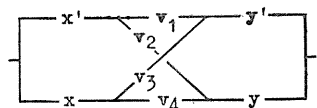


FIG. 5.2.9

ends of the circuit and sandwiches the coefficient part. In order to use this method, the four conditions:

$$\begin{aligned} x'y'v_2v_3v_4 &\leq f, & x'yv_1v_3v_4 &\leq f, \\ xy'v_1v_2v_4 &\leq f, & xyv_1v_2v_3 &\leq f \end{aligned}$$

should be satisfied, since, otherwise, a sneak path appears in the circuit. For example, when the condition  $x'y'v_2v_3v_4 \leq f$  is violated, the detour  $x' - v_2 - v_4 - v_3 - y'$  becomes a sneak path.

The sandwich method may not seem to be of much use because of the restricting conditions. But the fact is that the conditions are not so restrictive as they first seem to be and it is sometimes possible to make use of the prohibited detours with advantage (see Example 5.2.4).

**Example 5.2.3.** Synthesize a circuit with the transmission:

$$f = w'x'z' + w'xy' + wx'z + wxz.$$

**Solution.** The transmission function is already expanded with respect to  $w$  and  $x$ . The circuit synthesized by the sandwich method is shown in Fig. 5.2.10. No sneak path is present in the circuit, because the four conditions are satisfied by virtue of  $yy' = 0$  and  $zz' = 0$ . Since  $C = E = 8$ , the circuit is minimal in contacts.



*Example 5.2.4.* Synthesize a circuit with the transmission:

$$f = y'z'x + y'zw + yz'w + yz(w'x' + wx).$$

*Solution.* Omitting the last term  $yzwx$  for the moment, we construct a circuit for the remaining part of the function by the sandwich method. The result is shown in Fig. 5.2.11. In this circuit, the detour  $y-w-x-w-z$  is alive and realizes the omitted term  $yzwx$ . Hence, the circuit, as it stands, realizes the function  $f$ . It requires 9 contacts and 15 springs. It is probably minimal in contacts but not in springs, because a better circuit exists as will be shown in Example 5.2.6.

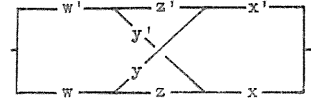


FIG. 5.2.10

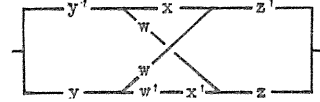


FIG. 5.2.11

### 5.2.3. Cross Connection Method

In this method, a transmission function  $f$  expanded as

$$f = x'y'v_1 + x'v_2 + xy'v_3 + xyv_4$$

is to be realized by one of the two circuits shown in Fig. 5.2.12. In any of the circuits, the expansion part is connected in the shape of a cross. Evidently, this method is applicable only when  $v_1v_3 \leq f$  and  $v_2v_4 \leq f$ , or  $v_1v_2 \leq f$  and  $v_3v_4 \leq f$  according as whether the circuit (a) or (b) is used.

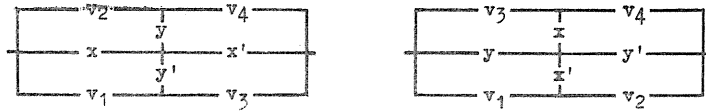


FIG. 5.2.12

This method is very powerful with a rather wide scope of application. Sometimes it is even possible to utilize the prohibited paths with profit.

*Example 5.2.5.* Synthesize a circuit with the transmission:

$$f = x \oplus y \oplus z = x'y'z + x'yz' + xy'z' + xyz.$$

*Solution.* The disjunctive tree method or the sandwich method, when applied to the function, yields the well-known circuit of Fig. 5.2.13. The cross connection method, on the other hand, yields the circuit of Fig. 5.2.14. This circuit is as good as that of Fig. 5.2.13 in the number of contacts and in the number of springs.

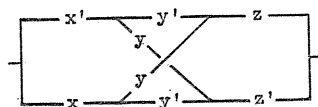


FIG. 5.2.13

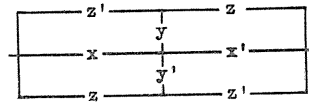


FIG. 5.2.14

*Example 5.2.6.* Synthesize a circuit with the transmission:

$$f = wx + wy'z + wyz' + xy'z' + w'x'yz.$$

*Solution.* The transmission function is the same as that of Example 5.2.4. The cross connection method, when applied to the function with the first term missing, yields the circuit of Fig. 5.2.15. Since this circuit includes the path  $wx$  automatically,

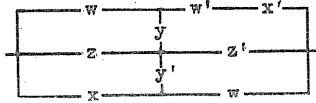


FIG. 5.2.15

it realizes the function  $f$  without any additional modification. It is as simple as the circuit of Fig. 5.2.11 both in contacts and in springs. Now we note that the  $x$  and the  $x'$  contacts are far apart and cannot be combined into a transfer contact. But it is possible to rearrange these two contacts so that they may be combined into a transfer contact. To begin with, we note that the term  $wy'z$  may be replaced by  $wx'y'z$  without affecting the transmission. Therefore, in trial, we insert a contact  $x'$  in series to the contact  $w$  in the right lower side as shown in Fig. 5.2.16. This circuit does not realize  $f$  because the path  $wx$  is now dead. However, by moving the two  $x'$  contacts to the left of  $z$  contact and merging them into a single contact, the path  $wx$  reappears.

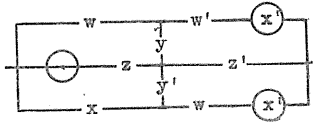


FIG. 5.2.16

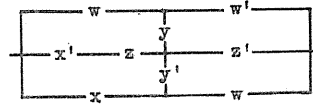


FIG. 5.2.17

In the modified circuit of Fig. 5.2.17, the  $x$  and  $x'$  contacts share the left terminal and hence can be combined into a transfer contact. This circuit requires 9 contacts and 14 springs, and is simpler by one spring than the circuit of Fig. 5.2.11.

The modification was enabled by replacing the term  $wy'z$  with a longer term  $wx'y'z$ . This tells us that, although the minimal sum is usually the best form of transmission function to use in the synthesis, it is not always advisable to adhere to the minimal sum.

#### 5.2.4. Square Connection Method

Assume that the given transmission function  $f$ , when expanded with respect to two variables, say,  $x$  and  $y$ , takes the form:

$$f = (x'y' + xy)g + (x'y + xy')h.$$

The square connection method is to realize such a function by one of the circuits of Fig. 5.2.18. The four contacts of the expansion part are now connected in the shape of a square loop. Obviously, it is necessary to take  $g_0$ ,  $g_1$ ,  $h_0$  and  $h_1$  so that  $g_0g_1 = g$  and  $h_0h_1 = h$ , and,  $x'g_0h_1 \leq f$  and  $xg_1h_0 \leq f$ , or,  $y'g_0h_1 \leq f$  and  $yg_1h_0 \leq f$ , according as the circuit (a) or (b) is used. This method has the following two drawbacks. That is: Its scope of application is very narrow and the

four contacts of the expansion part cannot be combined into transfer contacts. Nevertheless, when it is applicable, it brings forth a very good circuit which cannot be obtained by other methods.

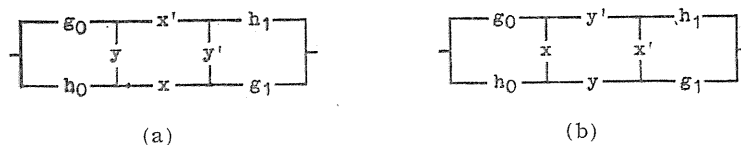


FIG. 5.2.18

*Example 5.2.7.* Synthesize a circuit with the transmission:

$$f = x \oplus y \oplus z = x'y'z + x'yz' + xy'z' + xyz.$$

*Solution.*  $f$  can be rewritten as:

$$f = (x'y' + xy)z + (x'y + xy')z',$$

so we may take  $g_0 = g_1 = z$  and  $h_0 = h_1 = z'$ . The result of the synthesis is given in Fig. 5.2.19. This circuit requires 8 contacts as the previous two circuits, but requires 14 springs. In this respect, it is worse than the previous ones.

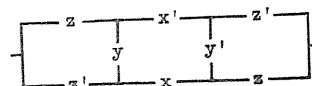


FIG. 5.2.19

*Example 5.2.8.* Synthesize a circuit with the transmission:

$$f = w'x'y'z' + w'x'yz + wxy'z + wxyz'.$$

*Solution.* Expanding  $f$  with respect to  $y$  and  $z$ , we obtain

$$f = (y'z' + yz)w'x' + (y'z + yz')wx.$$

An appropriate choice of  $g_0$ ,  $g_1$ ,  $h_0$  and  $h_1$  is now  $g_0 = w'$ ,  $g_1 = x'$ ,  $h_0 = x$  and  $h_1 = w$ . Thus the circuit of Fig. 5.2.20 will be obtained. This circuit is minimal in contacts because of  $C = E = 8$ , and no circuit which is minimal in contacts has been found yet other than this or those similar to this. But it requires as many as 16 springs, and it is not minimal in springs. In fact, a circuit requiring 10 contacts and 15 springs can be synthesized by the disjunctive tree method as shown in Fig. 5.2.21.

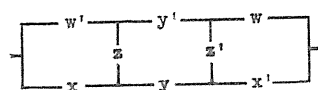


FIG. 5.2.20

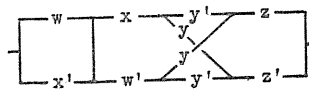


FIG. 5.2.21

### 5.3. Absorption Method<sup>1)</sup>

The successive steps of the absorption method are described as follows.

(1) Expand the given transmission function  $f$  by one of the variables. For the sake of determinateness, let us put

$$f = x'g + xh.$$

(2) Construct a circuit as shown in (a) of Fig. 5.3.1.

(3) Modify the two residue networks  $g$  and  $h$  until they will become identical. A general stratagem for the modification is to cut open a solid connection and insert a contact which is the same as the external contact into the gap and to connect a pair of nodes with a contact which is complementary to the external contact. These procedures are valid by virtue of Theorem 2.1.3. When the residue networks are alike, the modification is usually very simple. But, when they are not alike, the imagination will be required for finding out the necessary modification.

(4) Superpose the two identical residue networks and discard the external contacts as shown in (b) and (c) of Fig. 5.3.1. The resulting circuit realizes  $f$  automatically. In effect, we absorb the external contacts into the residue networks.

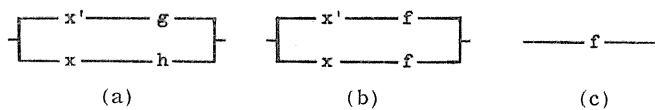


FIG. 5.3.1

*Example 5.3.1.* Synthesize a circuit with the transmission:

$$f = vw + xy + vyz + wxz.$$

*Solution.* Expanding  $f$  with respect to  $z$ , we obtain

$$f = z(v + x)(w + y) + z'(vw + xy).$$

We construct the series-parallel circuit of Fig. 5.3.2. The residue networks are very alike. They contain identical contacts in the same positions. The only difference between them is that the  $z$ -residue has a solid connection between the nodes 1 and 2 but the  $z'$ -residue has no connection between the nodes 3 and 4. Thus, in order to make the residue networks identical, we may replace the solid connection between the nodes 1 and 2 with a  $z$  contact and connect the nodes 3 and 4 with a  $z$  contact. The modified network is the planar bridge of Fig. 5.3.3. Of course this circuit is minimal in contacts.

In the above example, only external contacts were used to modify the residue networks. In general, however, other contacts must be used together with the external contacts. The following example illustrates such a situation.

<sup>1)</sup> cf. Ref. 4, p. 287.

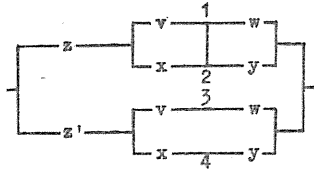


FIG. 5.3.2

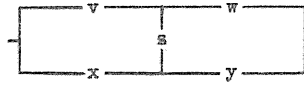


FIG. 5.3.3

*Example 5.3.2.* Synthesize a circuit with the transmission:

$$f = wxy + wyz' + xy'z + w'x'yz.$$

*Solution.* Expanding  $f$  with respect to  $x$ , we obtain

$$f = x(wy + y'z) + x'y(wz' + w'z).$$

We construct the residue networks as shown in (a) of Fig. 5.3.4. The networks are not alike. The only similarity is that they contain the  $w$  and  $z$  contacts in the same positions. They contain the  $y$  contacts but the positions are not exactly the same. Accordingly, we must first relocate the  $y$  contacts into the same position. This is done by adding an  $x$  contact to the right terminal of the  $x$ -residue and by inserting an  $x'$  contact in series to the  $y$  contact of the  $x'$ -residue. The modified networks are shown in (b) of Fig. 5.3.4.

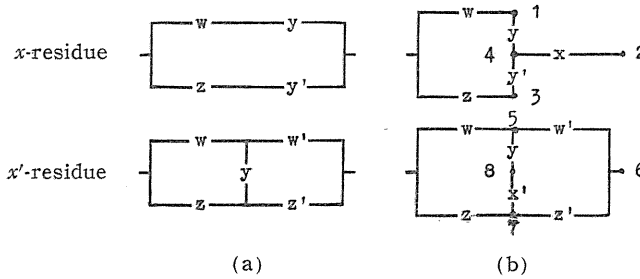


FIG. 5.3.4

Now the two residue networks must be turned into an identical network. It is not difficult to find out that the necessary procedure is to insert a  $w'$  contact between the nodes 1 and 2, a  $z'$  contact between the nodes 2 and 3, an  $x'$  contact between the nodes 3 and 4, an  $x$  contact between the nodes 6 and 8, and a  $y'$  contact between the nodes 7 and 8. Thus, the circuit of Fig. 5.3.5 will be obtained. This circuit is minimal both in contacts and springs.

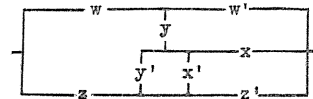


FIG. 5.3.5

#### 5.4. Path Accumulation Method<sup>1)</sup>

The general idea of the path accumulation method is to realize a transmission

<sup>1)</sup> cf. Ref. 4, p. 301.

function usually given as the minimal sum not at once but successively term by term. At each step, we try to add another path realizing a remaining term of the function. An important point here is to use as many existing contacts as possible without introducing a sneak path. Sometimes we encounter a difficult situation where the contacts we are going to use form a closed loop. The difficulty will be overcome by cutting open the loop somewhere and inserting a harmless contact or contacts in the gap.

*Example 5.4.1.* Synthesize a circuit with the transmission:

$$f = w'x + w'y + xy'z' + x'y'z + x'y'z.$$

*Solution.* Starting with the first two terms, we construct the circuit shown in (a) of Fig. 5.4.1. The next two terms can be written in the form of  $z'(x' + y')(x + y)$ , and so can be realized as shown in (b) of Fig. 5.4.1. The last term  $x'y'z$  is now to be taken up. We want to make use of the existing  $x'$  and  $y'$  contacts. For this purpose, we modify the circuit as follows: First, the positions of  $z'$  and  $x' + y'$  are interchanged. Then, one of  $x'$  or  $y'$ , say,  $y'$  is cut off from the left terminal and a harmless contact  $z'$  is inserted into the gap. The modified circuit is shown in (c) of Fig. 5.4.1. Now it is easy to add the path  $x'y'z$ . We may only insert a  $z$  contact between the right terminal and the node 1. Thus, the circuit of (d) of Fig. 5.4.1 is obtained. This circuit requires 8 contacts and is probably minimal in contacts.

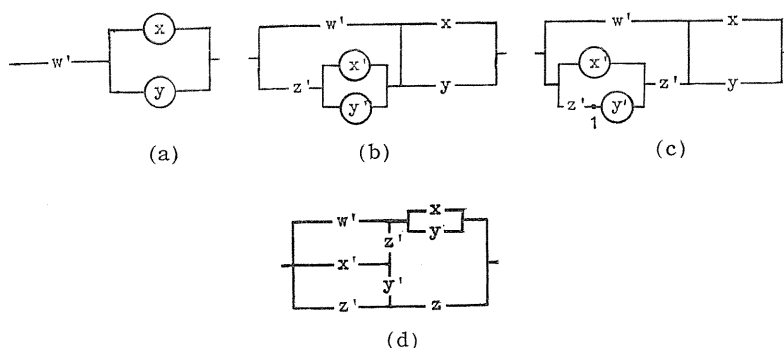


FIG. 5.4.1

The path accumulation method is very flexible; it places no preassumption for the final form of circuits and no restriction on the order of selection of terms of the transmission function. This flexibility is rather a disadvantage than an advantage of the method, because it means the lack of policy. A good plan which can serve as a remedy for the disadvantage is to endeavor to grasp the global structure of the transmission function by some means, for example, by Karnaugh maps. When we use this plan, it is not seldom that we can make use of good ready-made circuits.

Now, before working out some examples, we shall present several types of four-variable functions which have very good circuits as the building blocks for

other circuits and can be easily recognized by their appearances in Karnaugh maps.

The type shown in Fig. 5.4.2 may be called the *square* type, because each function of this type, when plotted in a Karnaugh map, takes the form of a square or a diagonal. Functions of the square type have two kinds of circuits, each of which is minimal both in contacts and springs. The typical circuits shown in Fig. 5.4.2 are both for the function plotted in (a).

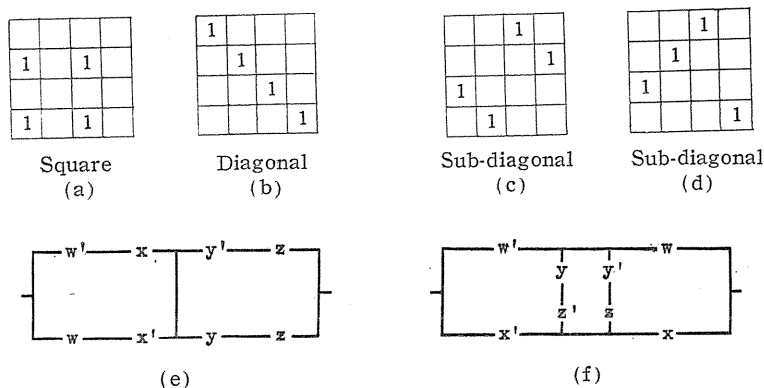


FIG. 5.4.2

The type shown in Fig. 5.4.3 may be called the *parallelogram* type, because each function of this type, when plotted in a Karnaugh map, takes the form of a parallelogram or an oblique square. The typical circuit shown in (d) is for the function (a), and is essentially the same as that of Fig. 5.2.20.

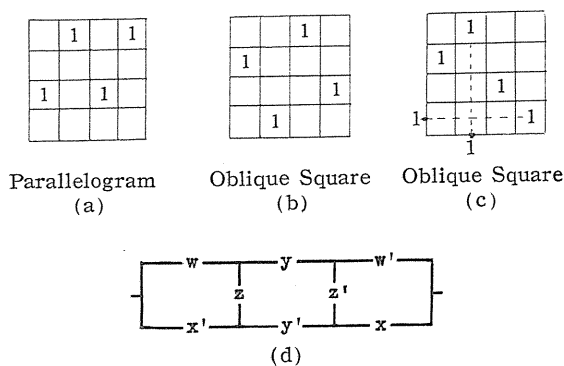


FIG. 5.4.3

The type shown in Fig. 5.4.4 may be called the *T* type, because each function of this type, when plotted in a Karnaugh map, takes the form of a letter "T" or a Japanese character "テ". The typical circuit shown in (c) is for the function (a).

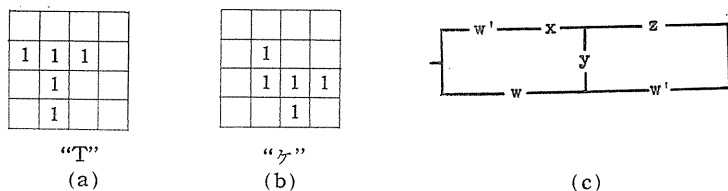


FIG. 5.4.4

The type shown in Fig. 5.4.5 may be called the *6-bridge*, because each function of this type is of the dimension 6 and its best circuit is a bridge like that of (d) for the function (a). The key point for recognizing this type by the Karnaugh map is to note that four 1-cells form a two-literal subcube and the remaining two are combined to two adjacent cells of the subcube.

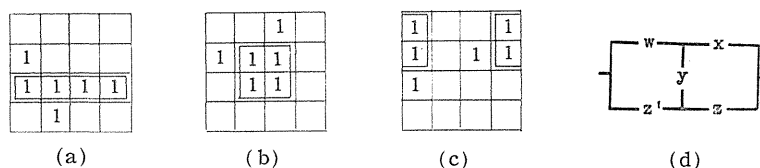


FIG. 5.4.5

The type shown in Fig. 5.4.6 may be called the *8-bridge*, because each function of this type is of the dimension 8 and its best circuit is a bridge like that of (d) for the function (a). The key point for recognizing this type by the Karnaugh map is to note that seven 1-cells are included in a one-literal subcube and the distance between the remaining 1-cell and the vacant cell of the subcube is 4. In the figures, the 1-cell which is not included in a one-literal subcube and the vacant cell of the subcube are marked with small circles.

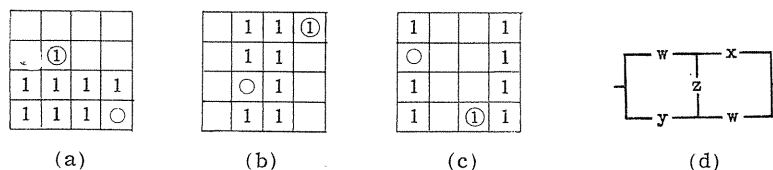


FIG. 5.4.6

Now we shall work out some examples illustrating the usefulness of the above mentioned plan.

*Example 5.4.2.* Synthesize a circuit with the transmission:

$$f = wxy + wxz + wyz + wx'y'z' + w'xy'z'.$$

*Solution.* Examining the Karnaugh map of  $f$ , we decompose  $f$  into two parts as



$$f = g + h,$$

where

$$g = w'xy'z' + wx'yz + wx'y'z'$$

and

$$h = wxy + wxz.$$

The Karnaugh maps of  $f$ ,  $g$  and  $h$  are shown in Fig. 5.4.7. The function  $g$  is, as it were, of an incomplete square type, because, if the vacant cell with the cross mark is filled with 1, it becomes a function of the square type.

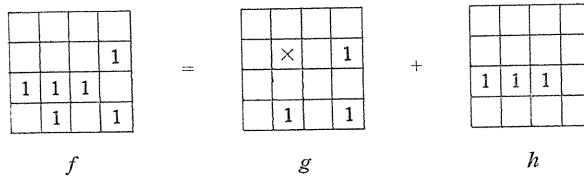


FIG. 5.4.7

Now we construct the circuit for the supplemented  $g$  as shown in Fig. 5.4.8 (a). The circuit for  $g$  can be easily obtained from this circuit by inserting a  $w$  contact between the  $y$  and  $z$  contacts, because the unnecessary path  $w'xyz$  corresponding to the cell with the cross mark is eliminated by this procedure. Next, the remaining part  $h$  is to be realized by using the existing contacts  $w, x, y$  and  $z$ . This can be done easily by inserting an  $x$  contact between the right terminal and the node 1. The final circuit shown in (c) requires 10 contacts and 16 springs. Probably this circuit is one of the minimal circuits. It is instructive to note that the inserted  $w$  contact plays the double role of eliminating the unnecessary path  $w'xyz$  and of facilitating the addition of the path  $wxz$ .

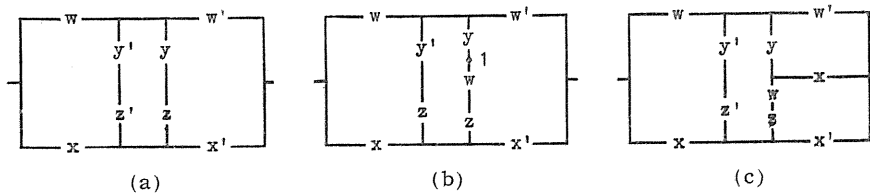


FIG. 5.4.8

**Example 5.4.3.** Synthesize a circuit with the transmission:

$$f = wxy + wx'y' + w'x'y + w'xz + w'y'z'.$$

**Solution.** It is not difficult to find out from the Karnaugh map that  $f$  can be decomposed into two parts  $g$  and  $h$  both of which are of the T type. The decomposition is shown in Fig. 5.4.9.

The functions  $g = wxy + w'xz + w'yz$  and  $h = wx'y' + w'x'z' + w'y'z'$  are realized by the circuits of (a) of Fig. 5.4.10. These two circuits are now to be combined into a single circuit. This can be done by merging the bottom line of the circuit  $g$  and the top line of the circuit  $h$ . Thus we obtain the circuit of (b). This

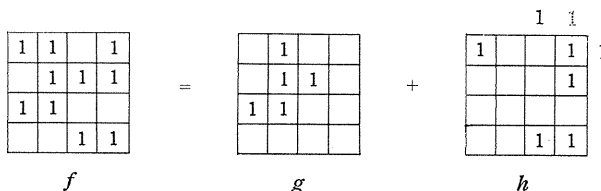


FIG. 5.4.9

circuit can be simplified further by merging two  $w'$  contacts in the left end into a single contact. The final circuit of (c) is a nonplanar bridge requiring 9 contacts and 14 springs, and is presumably minimal both in contacts and springs.

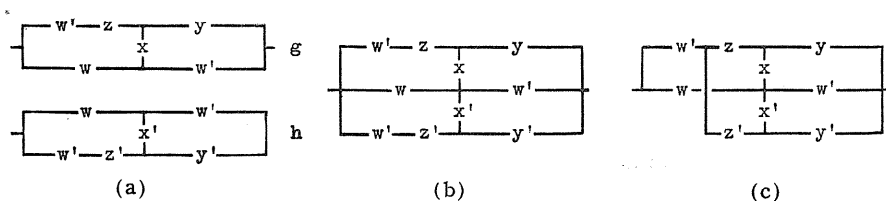


FIG. 5.4.10

**Example 5.4.4.** Synthesize a circuit with the transmission:

$$f = x'z' + w'y'z' + w'x'y + wxz + wx'y'.$$

**Solution.** There are two good ways to decompose  $f$  into two parts. One way is to decompose  $f$  as shown in Fig. 5.4.11.

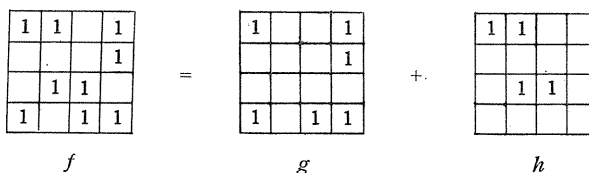


FIG. 5.4.11

The function  $g = x'z' + w'y'z' + wx'y'$  is of the 6-bridge type and can be realized by the circuit of (a) of Fig. 5.4.12. Two paths corresponding to the terms of  $h = wxz + w'x'y$  can be easily added by the existing  $x'$  and  $w$  contacts as shown in (b). The final circuit requires 9 contacts and 15 springs.

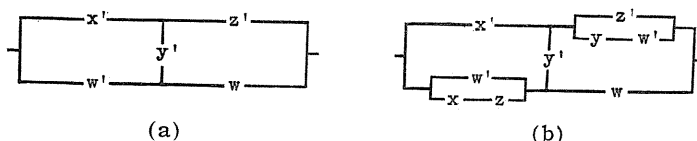


FIG. 5.4.12

Another good way to decompose  $f$  is shown in Fig. 5.4.13.

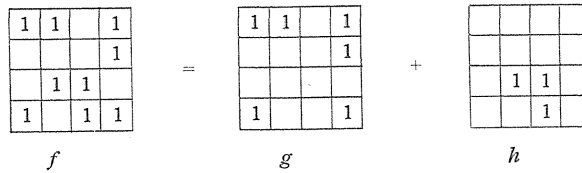


FIG. 5.4.13

The function  $g = x'z' + w'y'z' + w'x'y$  is also of the 6-bridge type and has the circuit shown in (a) of Fig. 5.4.14. The remaining part  $h = wxz + wy'z$  is best realized by adding the corresponding circuit independently of the circuit of  $g$ . The final circuit shown in (b) requires 9 contacts and 15 springs, and so is as good as the previous one.

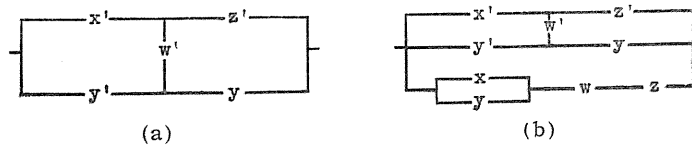


FIG. 5.4.14

We shall now show that we can synthesize a simpler circuit. First, we modify the circuit of (a) by inserting a  $z$  contact in series to the  $y$  contact as shown in (a) of Fig. 5.4.15. This modification does not affect the transmission of the circuit. Then, we add a path  $wy'z$  by inserting a  $w$  contact in parallel to the  $y$  contact as shown in (b). The term  $wxz$  is now to be added. Available existing contacts are  $w$  and  $z$ . After a few trials, it will be found that we may insert an  $x$  contact together with a  $w$  contact in parallel to the  $y'$  contact. The  $w$  contact is necessary to prevent the sneak path  $w'xz'$ . The final circuit shown in (c) requires 9 contacts and 14 springs, and so is simpler by one spring than the previous ones.

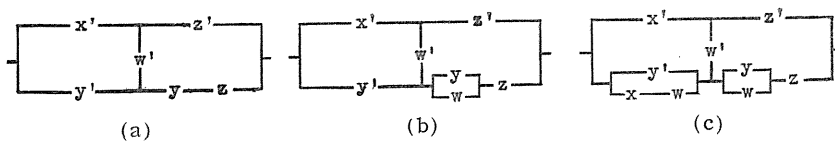


FIG. 5.4.15

### 5.5. Double Complementation Method

In the synthesis of relay switching circuits, we often encounter the situation that the given transmission function is complicated but its complement is comparatively simple. In such a situation, a reasonable method is first to synthesize a complementary circuit and then to apply the well-known method of geometrical

complementation to the complementary circuit. The method is called the *double complementation method*, because the final circuit is obtained as the geometrical complement of the complementary circuit. By this indirect method, it is frequently possible to obtain a good circuit which may be very difficult to synthesize by direct methods.

Here, two important points must be stated. One is that, between a circuit and its geometrical complement, the numbers of contacts are the same but the numbers of springs are generally different. The other is that the complementary circuit to be synthesized in the first step must be planar, because the method of geometrical complementation is applicable only to a planar circuit. This unfavorable fact is a practical limitation to the usefulness of the method.

Now we shall sketch the method of geometrical complementation without the proof. Suppose we are given a planar circuit with two terminals in the same horizontal line. To begin with, we place a node in the area enclosed in each mesh of the circuit, considering the top and the bottom spaces as meshes. Then we choose any pair of nodes and connect them with as many lines as the contacts located in the branch of the circuit separating the corresponding pair of areas. Each line is to be so drawn that it passes a contact when it crosses the separating branch. In each line thus drawn, we insert a complementary contact of the one passed by it. When the same procedure is repeated for every pair of nodes, the resulting circuit is the geometrical complement of the original circuit. Precisely speaking, it has the complementary transmission between the top and the bottom nodes.

The method of geometrical complementation is very powerful and requires no materials other than the circuit diagram. It can be applied to any planar circuit, but is not applicable to nonplanar circuits. The procedures may be cumbersome for beginners, but, after some experiences, all the work can be carried out mentally and the geometrical complement will be drawn immediately.

Let us now apply the method to the circuit shown in (a) of Fig. 5.5.1. The result is the circuit of (b). Note that the numbers of springs are different between the two circuits. The circuit (a) has 16 springs but the circuit (b) has 17 springs.

Now we shall work out some examples illustrating the merit of the double complementation method.

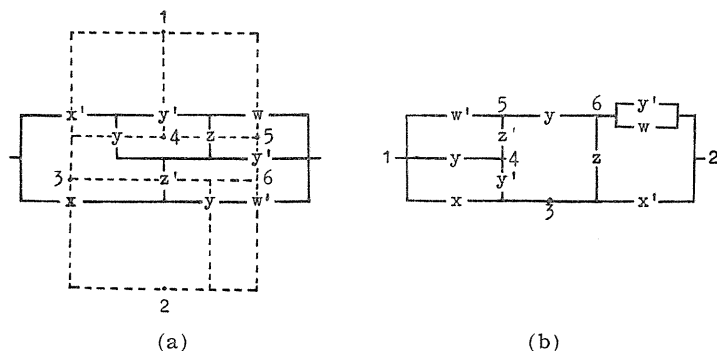


FIG. 5.5.1

*Example 5.5.1.* Synthesize a circuit with the transmission:

$$f = wxy + wyz' + xy'z + w'x'y'z'.$$

*Solution.* This function is rather simple but its complement

$$f' = w'y + x'z + wy'z' + xy'z'$$

is still simpler. This function is very easy to handle and can be realized by the commonplace circuit of (a) of Fig. 5.5.2. We now apply the method of geometrical complementation to this circuit. The result is shown in (b). This circuit is obviously minimal in contacts and in springs. It would be very difficult to obtain such a good circuit as this by other methods.

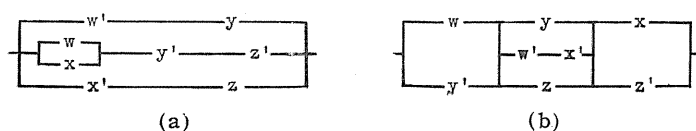


FIG. 5.5.2

*Example 5.5.2.* Synthesize a circuit with the transmission:

$$f = wxy' + wx'z + wyz' + w'xyz + w'x'y'z'.$$

*Solution.* This time the double complementation method loses its proper function, because  $f$  is self-complementary or  $f$  is congruent to its own complement. Accordingly, we cannot help resorting to a direct method. The Karnaugh map shown in Fig. 5.5.3 suggests that a reasonable plan here is to decompose  $f$  into two parts and use the path accumulation method. The appropriate decomposition is as follows:

$$f = g + h,$$

where

$$g = w'x'y'z' + w'xyz + wx'y'z' + wxy'z$$

and

$$h = wx'z + wxz'.$$

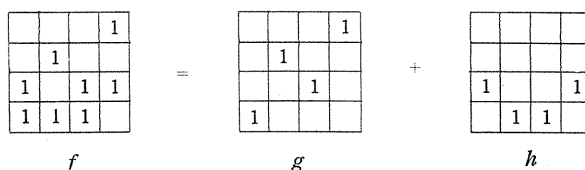


FIG. 5.5.3

The function  $g$  is of the parallelogram type and can be realized by the circuit shown in (a) of Fig. 5.5.4. A path  $wx'z'$  will appear when we interchange the  $x'$  and  $z'$  contacts. This modification does not affect the four paths of  $g$  but it gives rise to a sneak path  $w'x'z$  along the bottom line. Thus, inserting a harmless  $z'$  contact in series to the  $x'$  contact to eliminate the sneak path, we obtain the circuit of (b). Now a path  $wx'z$  must be added. We can do this by inserting a

$w$  contact between the nodes 1 and 2 together with a  $z$  contact which prevents the appearance of a sneak path  $wx'y'z'$ . The final circuit is shown in (c). The circuit requires 11 contacts and 18 springs.

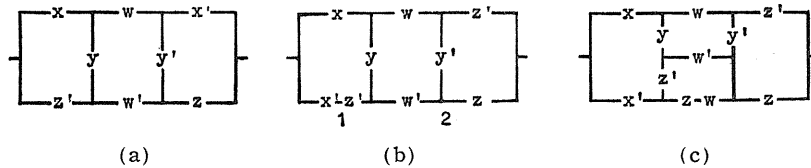


FIG. 5.5.4

Here we note that  $f'$  can be obtained from  $f$  by complementing  $w$ , since  $f$  can be written as

$$f = w \oplus (xyz + x'y'z').$$

Hence, by interchanging the  $w$  and  $w'$  contacts in the circuit of Fig. 5.5.4, a complementary circuit will be obtained. The method of geometrical complementation, when applied to the circuit, yields another circuit for  $f$  shown in Fig. 5.5.5. This circuit requires 11 contacts and 17 springs, and so is simpler by one springs than the previous one.

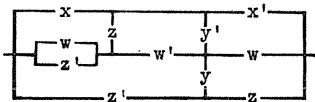


FIG. 5.5.5

In general, when we have synthesized a planar circuit for a self-complementary transmission function, another circuit can easily be obtained from it by the double complementation method, where the first complementation will be done by an appropriate symmetry and the second complementation by the method of geometrical complementation.

## 5.6. Synthesis of Circuits for Symmetric Functions

In the synthesis of circuits for symmetric functions, the standard method is to use the well-known *symmetric trees*. A symmetric tree is a multi-terminal network realizing all the fundamental symmetric functions simultaneously. For example, the network shown in Fig. 5.6.1 is the symmetric tree for five variables. Each branch of the network represents a contact. Specifically, a horizontal branch represents a break contact and an oblique branch a make contact. The contacts located in the same vertical line have the same designation. Thus, for example, the four horizontal branches and the four oblique branches in the column identified as  $y$  represent the contacts  $y'$  and  $y$  respectively. The numbers attached to

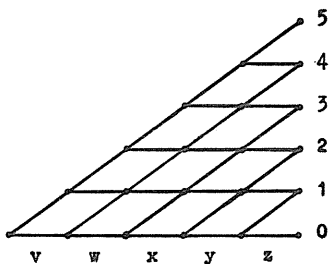


FIG. 5.6.1

the right terminals represent the  $a$ -numbers. For example, the number 2 means that the transmission between the left terminal and the terminal 2 is given by

the fundamental symmetric function  $S_2(v, w, x, y, z)$ .

In order to realize a given symmetric function, it suffices to connect the terminals corresponding to the relevant  $\alpha$ -numbers together and delete the redundant part. Under most circumstances, however, further simplifications are possible by other methods.

*Example 5.6.1.* Synthesize a circuit with the transmission:

$$f = S_{0,1}(v, w, x, y, z).$$

*Solution.* The relevant part of the symmetric tree for this problem is shown in Fig. 5.6.2. When we combine the terminals 0 and 1, the encircled  $z$  contact and  $z'$  contact can be replaced with a solid connection by virtue of  $z + z' = 1$ . In general, such a simplification as this is possible when we combine the consecutive terminals. A similar simplification is possible at the left end in the present case, when we add one more path  $v'w'x'y'z'$  along the top side by inserting a  $v'$  contact in parallel to the  $v$  contact. The final circuit is shown in Fig. 5.6.3.

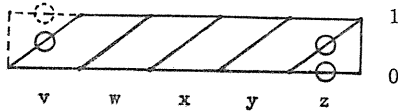


FIG. 5.6.2

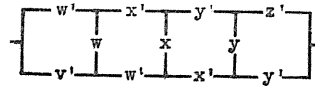


FIG. 5.6.3

*Example 5.6.2.* Synthesize a circuit with the transmission:

$$f = (v \oplus w \oplus x \oplus y \oplus z)' = S_{0,2,4}(v, w, x, y, z).$$

*Solution.* The relevant part of the symmetric tree for this problem is shown in Fig. 5.6.4. First, we combine the terminals 4 and 2. Then, the encircled two  $z'$  contacts can be merged into a single contact as shown in Fig. 5.6.5. In effect, the upper part beyond the level 3 is *folded* down onto the lower part.

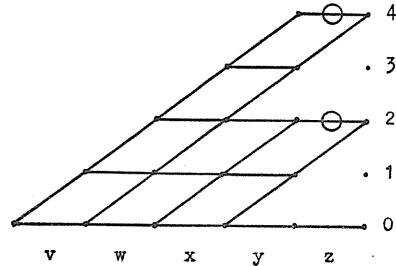


FIG. 5.6.4

Further, it is observed that the two identical parts each consisting of the three contacts with the marks  $\bigcirc$ ,  $\ominus$  and  $\oplus$  can be merged into a single part. This is done also by folding the upper part beyond the level 3 is *folded* down onto the lower part as shown in Fig. 5.6.5.

Now the terminals 0 and 2, 4 must be combine. When they are combined, it is again possible to merge the two identical parts each consisting of the five contacts with the marks  $\bigcirc$ ,  $\ominus$ ,  $\oplus$ ,

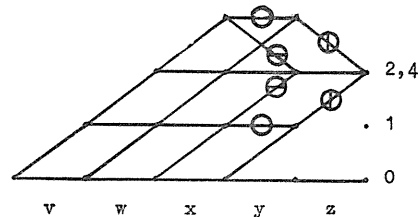


FIG. 5.6.5

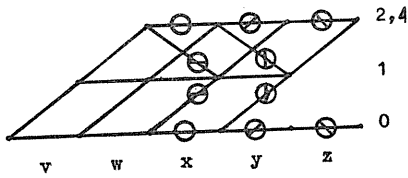


FIG. 5.6.6

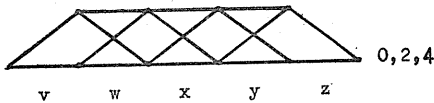


FIG. 5.6.7

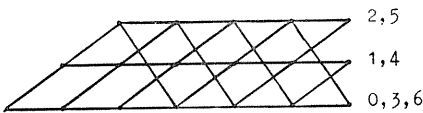


FIG. 5.6.8

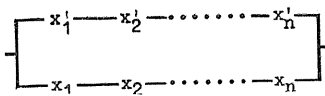


FIG. 5.6.9

without affecting the transmission of the whole circuit. In this new form two pairs of complementary contacts can be combined into transfer contacts. When the same deformation is carried out in every similar part of the circuit, our purpose will be accomplished. Thus, for example, the function  $S_{0,5}(v, w, x, y, z)$  will be realized by the circuit of Fig. 5.6.11, and the function  $S_{0,6}(u, v, w, x, y, z)$  by either of the circuits of Fig. 5.6.12.

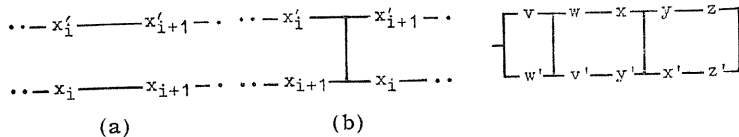


FIG. 5.6.10

FIG. 5.6.11

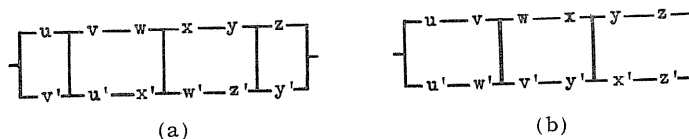


FIG. 5.6.12

⊙ and ⊙ into a single part. Thus, in effect, the upper half is folded down onto the lower half. The result is the well-known circuit shown in Fig. 5.6.7.

In general, when the  $a$ -numbers form an arithmetic progression beyond or beneath some number, the folding procedures can be used with profit. For example, the network shown in Fig. 5.6.8 is synthesized by the folding procedures. It is a network realizing the three functions  $S_{0,3,6}$ ,  $S_{1,4}$  and  $S_{2,5}$  simultaneously.

*Example 5.6.3.* Synthesize a circuit with the transmission:

$$f = S_{0,n}(x_1, x_2, \dots, x_n).$$

*Solution.* This problem is easily solved by the circuit of Fig. 5.6.9. The circuit is obviously minimal in contacts but it is uneconomical in regard to the number of transfer contacts. Now, in order to improve this point, we consider a part of the circuit consisting of two consecutive contacts such as shown in (a) of Fig. 5.6.10. This part can be deformed as shown in (b)



5.7. Boolean Matrix Method<sup>1)</sup>

Suppose we are given a relay contact network. To begin with, we select all the terminal nodes and some of non-terminal nodes of the network, and number them in an arbitrary manner. Then, we construct a matrix with as many rows and columns as the selected nodes, and enter the direct transmission from the node  $i$  to the node  $j$  into  $ij$ -position. This matrix is called a *connection matrix*. Evidently, every diagonal entry is 1 in every connection matrix. Further, a connection matrix is symmetric when the network contains no unilateral element such as a rectifier, since then the transmission from the node  $i$  to the node  $j$  is identical with the transmission from the node  $j$  to the node  $i$ . Hereafter, we assume that every connection matrix is symmetric.

A special connection matrix where all the nodes are taken into account is called the *primitive connection matrix*. In such a matrix, every entry is 0, 1, or a single literal or a sum of literals. For example, the primitive connection matrix of the two-terminal network of Fig. 5.7.1 is given by the following matrix.

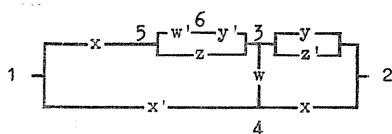


FIG. 5.7.1

	1	2	3	4	5	6
1	1	0	0	$x'$	$x$	0
2	0	1	$y+z'$	$x$	0	0
3	0	$y+z'$	1	$w$	$z$	$y'$
4	$x'$	$x$	$w$	1	0	0
5	$x$	0	$z$	0	1	$w'$
6	0	0	$y'$	0	$w'$	1

On the other hand, the non-primitive connection matrix of the same network where the nodes 5 and 6 are excluded from the consideration is given by

	1	2	3	4
1	1	0	$x(z+w'y')$	$x'$
2	0	1	$y+z'$	$x$
3	$x(z+w'y')$	$y+z'$	1	$w$
4	$x'$	$x$	$w$	1

Another Boolean matrix called the *output matrix* is defined for any network. This matrix has as many rows and columns as the terminal nodes and its  $ij$ -entry is the total transmission from the terminal  $i$  to the terminal  $j$ . Thus, for example, the output matrix of the network of Fig. 5.7.2 is given by the following matrix.

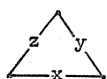


FIG. 5.7.2

$$\begin{bmatrix} 1 & z+xy & y+xz \\ z+xy & 1 & x+yz \\ y+xz & x+yz & 1 \end{bmatrix}$$

<sup>1)</sup> cf. Ref. 11.

Clearly any two-terminal network has an output matrix of the form

$$\begin{bmatrix} 1 & f \\ f & 1 \end{bmatrix}$$

Now, in terms of Boolean matrices, the analysis and the synthesis of a network are expressed as follows: The analysis is to derive the output matrix from a given connection matrix, and the synthesis is to derive a primitive or a nearly primitive connection matrix from a given output matrix.

A standard method for the analysis consists of the following steps.

(1) Write down a connection matrix  $C = (c_{ij})$ , primitive or non-primitive, corresponding to the given network, where  $c_{ij}$  is the  $ij$ -entry of  $C$ .

(2) Choose any number, say,  $r$  assigned to a non-terminal node. Construct a matrix  $(c_{ij} + c_{ir}c_{rj})$  and delete the  $r$ -th row and  $r$ -th column. The resulting matrix  $C_1$  is also a connection matrix of the network, because its  $ij$ -entry is the sum of the direct transmission  $c_{ij}$  and the indirect transmission  $c_{ir}c_{rj}$  by way of the node  $r$  between the nodes  $i$  and  $j$ . But it has no row and column corresponding to the non-terminal node  $r$ .

Apply the same procedure to  $C_1$  to obtain a connection matrix  $C_2$  where two of non-terminal nodes are excluded from consideration. Apply the same procedure to  $C_2$  to obtain a connection matrix  $C_3$  where three of non-terminal nodes are excluded from consideration, and so forth. Thus, we shall eventually arrive at a connection matrix  $C_0$  with no row and column corresponding to non-terminal nodes. We call this connection matrix as the *reduced matrix*.

(3) Calculate  $C_0^2, C_0^3, \dots$  etc. until we shall find that  $C_0^p = C_0^{p+1}$  for a certain  $p$ . Then,  $C_0^p$  is the output matrix we are seeking for. This step is based on the following two theorems due to Lunts<sup>1)</sup>, Hohn and Schissler.<sup>2)</sup>

*Theorem 5.7.1. Let  $A$  be any connection matrix with  $m$  rows and columns, then there exists an integer  $p$  such that  $p \leq m-1$  and  $A \leq A^2 \leq \dots \leq A^p = A^{p+1} = \dots$*

*Theorem 5.7.2. Let  $A$  be the reduced matrix and  $F$  be the output matrix of an  $m$ -terminal network, then there exists an integer  $p$  such that  $p \leq m-1$  and  $A^p = F$ .*

Thus, the analysis is reduced to a routine calculation of Boolean matrices. The synthesis, on the contrary, is very complicated. There is no standard method like that of the analysis. Perhaps, the most powerful tool for the synthesis will be the *node insertion*, the reverse procedure of the node elimination used in the second step of the standard method of the analysis. Precisely speaking, the node insertion means to enlarge a connection matrix or an output matrix of the form  $(c_{ij} + c_i c_j)$  by adding a new row  $(c_1, c_2, \dots, 1)$  and a new column  $(c_1, c_2, \dots, 1)$  and eliminate the terms  $c_i c_j$  from the existing entries. Another important tool is the manipulation of redundant terms. Sometimes a connection matrix may be simplified by eliminating redundant terms, and other times redundant terms may be added to some entries so that the node insertion may become applicable.

Now we shall explain how to use these tools by the following examples.

<sup>1)</sup> cf. Ref. 17. <sup>2)</sup> cf. Ref. 11.

*Example 5.7.1.* Synthesize a circuit with the transmission:

$$f = wxy' + wx'y + xyz.$$

*Solution.* The output matrix is given by

$$\begin{bmatrix} 1 & wxy' + wx'y + xyz \\ wxy' + wx'y + xyz & 1 \end{bmatrix}$$

First, by inserting a new node 3, the terms  $wxy'$  and  $wx'y$  of the 12-entry and 21-entry can be eliminated as follows.

$$\begin{bmatrix} 1 & xyz & w \\ xyz & 1 & x'y + xy' \\ w & x'y + xy' & 1 \end{bmatrix}$$

It should be noted here that the node insertion can be applied to any output matrix with two rows and columns. Now we try to eliminate the terms  $xyz$  and  $x'y$  by inserting a new node 4. Since the two terms are in the second row (column) and share the factor  $y$ , the 24-entry (42-entry) of the new connection matrix must be  $y$ . Then, it follows that the 41-entry (14-entry) must be  $xz$  and the 43-entry (34-entry) must be  $x'$ . Further, in order that the node insertion may be valid, it is necessary that the 13-entry (31-entry) should include the product of the 14-entry and the 43-entry. In the present case, the 13-entry is  $w$  and the product of the 14-entry and the 43-entry is 0. Therefore the procedure is really valid. The result is given by

$$\begin{bmatrix} 1 & 0 & w & xz \\ 0 & 1 & xy' & y \\ w & xy' & 1 & x' \\ xz & y & x' & 1 \end{bmatrix}$$

This is not a primitive connection matrix but it leaves no room for further improvements. The circuit corresponding to the matrix is shown in Fig. 5.7.3. In this example, we have seen that, whenever there are two terms of the form  $xg$  and  $xh$  such that  $gh=0$  in a row (column) of a connection matrix, we are able to insert a node to eliminate these terms. Obviously, the procedure amounts to the saving of one  $x$  contact.

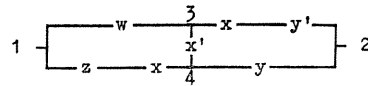


FIG. 5.7.3

*Example 5.7.2.* Synthesize a circuit with the transmission:

$$f = xyz + wx + wy + wz.$$

*Solution.* The output matrix is given by

$$\begin{bmatrix} 1 & x(yz + w) + wy + wz \\ x(yz + w) + wy + wz & 1 \end{bmatrix}$$

First, we eliminate the term  $x(yz+w)$  by inserting a node 3 as follows.

$$\begin{bmatrix} 1 & wy + wz & x \\ wy + wz & 1 & yz + w \\ x & yz + w & 1 \end{bmatrix}$$

Next, we want to eliminate the terms  $wz$  and  $yz$  by the node insertion. But this time the procedure is not valid because 13-entry does not include  $wy$ . Now it is observed that there is a path  $wy$  from the node 1 to the node 2, and a path  $w$  from the node 2 to the node 3, and hence, an indirect path  $wy$  from the node 1 to the node 3 by way of the node 2. Accordingly, we may add a direct path  $wy$  between the node 1 and the node 3. Thus, we obtain

$$\begin{bmatrix} 1 & wy + wz & x + wy \\ wy + wz & 1 & w + yz \\ x + wy & w + yz & 1 \end{bmatrix}$$

In this form, the node insertion above mentioned is possible. The result is given by

$$\begin{bmatrix} 1 & wy & x & w \\ wy & 1 & w & z \\ x & w & 1 & y \\ w & z & y & 1 \end{bmatrix}$$

Here, again, we note that there are a path  $w$  from the node 1 to the node 4, a path  $y$  from the node 4 to the node 3 and a path  $w$  from the node 3 to the node 2, and consequently, an indirect path  $wy$  from the node 1 to the node 2 by way of the nodes 4 and 3. Hence the 12-entry (21-entry)  $wy$  is redundant and can be deleted.

The simplified matrix is given by

$$\begin{bmatrix} 1 & 0 & x & w \\ 0 & 1 & w & z \\ x & w & 1 & y \\ w & z & y & 1 \end{bmatrix}$$

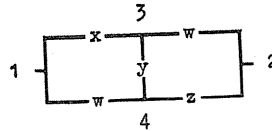


FIG. 5.7.4

This is a primitive connection matrix and represents the planar bridge of Fig. 5.7.4.

*Example 5.7.3.* Synthesize a circuit with the transmission:

$$f = wx'z' + wxy + w'y' + w'xz.$$

*Solution.* We write the output matrix as follows.

$$\begin{bmatrix} 1 & w(x'z' + xy) + w'(y' + xz) \\ w(x'z' + xy) + w'(y' + xz) & 1 \end{bmatrix}$$

Inserting two nodes 3 and 4 to eliminate all terms of 12- and 21-entries, we obtain

$$\begin{bmatrix} 1 & 0 & w & w' \\ 0 & 1 & x'z' + xy & y' + xz \\ w & x'z' + xy & 1 & 0 \\ w' & y' + xz & 0 & 1 \end{bmatrix}$$

Now we want to eliminate the terms  $xy$  and  $xz$  in the second row and the second column by the node insertion. In order to be able to do this, the term  $yz$  to be added to the 34- and 43-entries must be redundant. Although the term is really redundant, it is not easy to recognize the redundancy of the term, because there is no indirect path  $yz$  between the nodes 3 and 4. Presumably, the only way to achieve the purpose may be to add  $yz$  to 34- and 43-entries and examine all the possible paths between the nodes 1 and 2. The procedure is very cumbersome, because we must calculate the output matrix of the modified connection matrix after all. At any rate, when the redundancy of the term  $yz$  is proved somehow, and the node insertion is carried out, we obtain the following matrix.

$$\begin{bmatrix} 1 & 0 & w & w' & 0 \\ 0 & 1 & x'z' & y' & x \\ w & x'z' & 1 & 0 & y \\ w' & y' & 0 & 1 & z \\ 0 & x & y & z & 1 \end{bmatrix}$$

Let us now reconsider the same problem from the ordinary viewpoint. The second and the third connection matrix represents respectively the circuits (a) and (b) of Fig. 5.7.5.

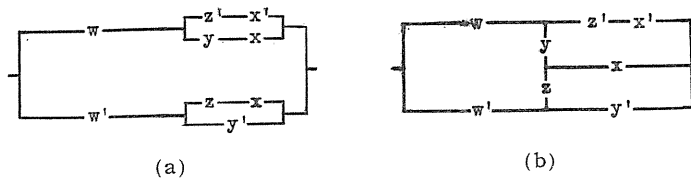


FIG. 5.7.5

The node insertion under consideration is equivalent to simplifying the circuit (a) by merging two  $x$  contacts into a single contact as shown in (b). This procedure is valid because, as will be seen easily, no sneak path appears in the circuit of (b). Thus, from the ordinary viewpoint, this problem is very easy, but, from the viewpoint of the Boolean matrix method, it is not easy.

The Boolean matrix method is drastically different from the other methods.

It has an aim to remove intuitive means such as circuit diagrams as far as possible and reduce the synthesis to a routine calculation of Boolean matrices. The aim has not so far been attained satisfactorily, but, in its rationality and especially in its direct applicability to multiterminal circuits, the Boolean matrix method is the most promising of all the methods.

## 6. Synthesis of Electronic Switching Circuits

### 6.1. Electronic Switching Circuits

In electronic switching circuits, each of the inputs and the outputs is specified by whether the voltage level is high or low, or by whether a pulse is present or absent at the corresponding terminal. In this paper, we shall concentrate our attention to one-output combinational switching circuits operating on voltage level. Nevertheless, most part of the following developments can be applied to similar circuits operating on pulse.

There are some essential differences between a relay switching circuit and an electronic switching circuit. Relay circuits are bilateral, but electronic circuits are unilateral. There are paths conducting electric current in a relay circuit, but there are no such paths in an electronic circuit. Relay circuits consist of relay contacts, each of which does not perform a logical operation by itself, but electronic circuits consist of *gates*, each of which performs a logical operation by itself.

The three gates performing addition, multiplication and complementation are fundamental. These gates are called *add gate*, *multiply gate* and *complement gate* respectively, and represented symbolically as shown in Fig. 6.1.1.

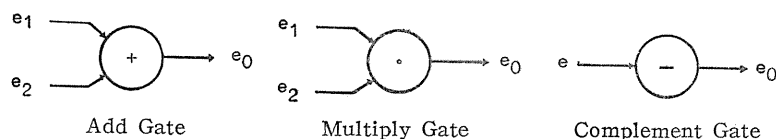


FIG. 6.1.1

The operations of these gates are given by the following table, where the values 0 and 1 signify the low and high voltage levels respectively.

By means of the three fundamental gates, every Boolean function of the

TABLE 6.1.1

$e_1$	$e_2$	$e_0$
0	0	0
0	1	1
1	0	1
1	1	1

$$e_0 = e_1 + e_2$$

Add Gate

$e_1$	$e_2$	$e_0$
0	0	0
0	1	0
1	0	0
1	1	1

$$e_0 = e_1 e_2$$

Multiply Gate

$e$	$e_0$
0	1
1	0

$$e_0 = e'$$

Complement Gate

input variables can be realized. For example, let us realize the function  $f = xyz + x'(y' + z')$ . Since  $f$  can be rewritten as  $f = xp + x'p' = (x + p')(x' + p)$  where  $p = yz$ , we obtain the following two circuits.

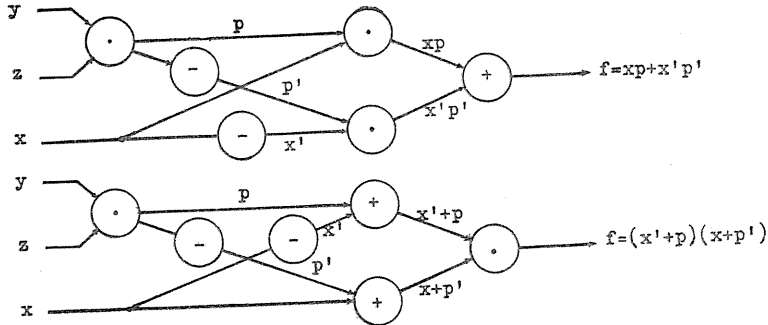


FIG. 6.1.2

Now we observe that in the upper circuit the input voltage  $x$  is used as the input to the complement gate producing  $x'$  and at the same time as the input to the multiply gate producing  $xp$ , and the voltage  $p$  is used as the input to the complement gate producing  $p'$  and at the same time as the input to the multiply gate producing  $xp$ . The same is observed in the lower circuit too. In general, a voltage represented by an input variable or a function of input variables can be used simultaneously as the inputs to arbitrarily many gates. Of course, there is a technical limitation on the number of gates to which a voltage can be applied simultaneously. Anyhow, this is an important property of electronic circuits. By virtue of this property, the substitution  $p = yz$  is very effective in the above example. Without this substitution, the circuits become more complicated. Obviously, one more reason why the substitution is successful is that  $f$  is functionally separable and can be written as  $f = x' \oplus yz$ . Thus, the recognition of the functional separability of Boolean functions is very important in the synthesis of electronic switching circuits. However substitutions are often useful even when the function to be realized is not functionally separable. We shall illustrate some of such situations by a few examples in later places.<sup>1)</sup>

## 6.2. Rectifier Switching Circuits

Many kinds of rectifiers are used for various purposes in switching circuits. For example, the unilateral nature of electric conduction of the rectifier can be used with profit in relay switching circuits. The following is an example of such a use of the rectifier. By virtue of the use of a rectifier as shown in Fig. 6.2.1, the function

$$f = w'x'y'z' + wxy'z' + wxyz$$

is realized with only 8 contacts. Without the rectifier, this function requires at

<sup>1)</sup> See pp. 164-167.

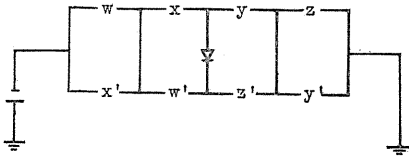


FIG. 6.2.1

least 9 contacts for its realization. Although such a use of the rectifier may be important, we here consider the rectifier as the component of electronic switching circuits.

The general construction of the rectifier add gate and the multiply gate is shown in Fig. 6.2.2.

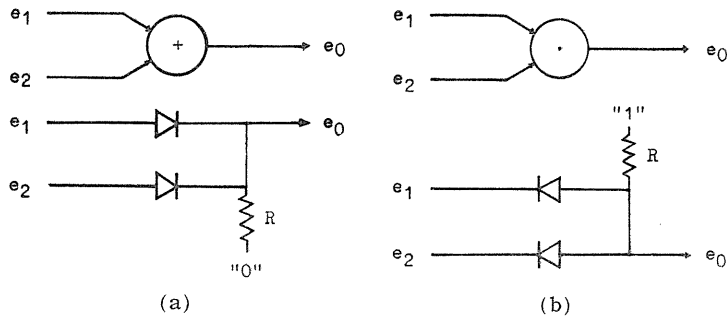


FIG. 6.2.2

In the gate of (a), the low voltage represented by 0 is applied to the output terminal through a resistor  $R$ , and two rectifiers are directed in the forward direction from the input terminals to the output terminal. When one of the input voltages is 1, the electric current flows through the corresponding rectifier and the resistor, and consequently, the output voltage becomes high. If we choose the resistance  $R$  sufficiently large compared with the forward resistance of the rectifiers, the output voltage is substantially as high as 1. When, on the other hand, both of the input voltages are 0, no current flows through the gate and the output voltage is 0. Thus, it has been shown that the gate (a) is really an add gate. By the similar consideration, it will be shown that the gate (b) is really a multiply gate. Obviously, these gates can be extended to gates with more than two inputs performing the logical operations  $e_0 = e_1 + e_2 + \dots + e_n$  and  $e_0 = e_1 e_2 \dots e_n$  respectively.

Complement gates cannot be formed by means of rectifiers. Accordingly, when a complement gate is required, we must use some other elements than rectifiers. However, when it is assumed that all the input variables and their complements are available as the input voltages, no complement gate is required and any Boolean function of the input variables can be realized by means of add gates and multiply gates. The assumption made above is not impractical. It will be satisfied usually when we are synthesizing a small part of a large apparatus. Therefore, we shall make this assumption hereafter whenever it is convenient.

Now we introduce the following two symbols.

$$F_n(e_1, e_2, \dots, e_n) = e_1 + e_2 + \dots + e_n$$

$$B_n(e_1, e_2, \dots, e_n) = e_1 e_2 \dots e_n$$



$F_n(e_1, e_2, \dots, e_n)$  represents an add gate and  $B_n(e_1, e_2, \dots, e_n)$  represents a multiply gate with  $n$  inputs  $e_1, e_2, \dots, e_n$ . Note that  $F$  stands for "forward" and  $B$  stands for "backward", and the suffix  $n$  indicates the number of inputs. With these symbols, any rectifier circuit can be represented conveniently by a symbolical expression. For example, the circuit shown in Fig. 6.2.3 can be represented by the expression

$$F_2[B_2(w, x), B_2(y', z)].$$

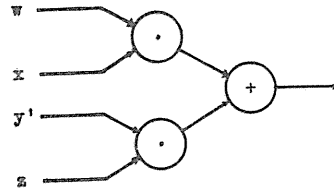


FIG. 6.2.3

Since this representation of rectifier circuits is very concise and the number of rectifiers required in a circuit can be easily obtained as the sum of suffices of all symbols, we shall use it exclusively hereafter.

Now suppose we have synthesized the circuit of Fig. 6.2.4 realizing the function  $f = z(y + wx)$ .

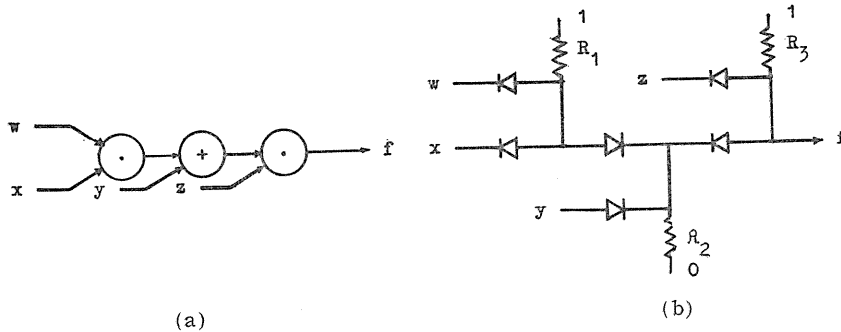


FIG. 6.2.4

In the circuit diagram (b), we see that the voltage level 1 is applied to the multiply gate producing  $wx$  and its output serves as one of the inputs of the add gate producing  $y + wx$ . The voltage level 0 is applied to the add gate through the resistor  $R_2$ . Accordingly, if  $w = x = 1$ , the resistors  $R_1$  and  $R_2$  are substantially in series between the voltages 1 and 0. In this state, the current flows from  $R_1$  to  $R_2$  and, in spite of  $wx = 1$ , the output of the multiply gate does not reach the level 1 but takes a certain lower value determined by the ratio  $R_2/R_1$ . In order to make this value nearly as high as the level 1, it is necessary to make  $R_2$  sufficiently large compared with  $R_1$ .

In the next stage of the circuit, the output of the add gate is one of the inputs of the final multiply gate. The voltage level 1 is applied to the gate through the resistor  $R_3$ . Thus, if both inputs of the add gate are in the 0 level, the resistors  $R_2$  and  $R_3$  are substantially in series between the voltages 1 and 0. In this state, the current flows from  $R_3$  to  $R_2$ , and, in spite of  $y + wx = 0$ , the output of the add gate does not go down to the level 0 but takes a certain higher value determined by the ratio  $R_3/R_2$ . In order to make this value nearly as low as the level 0, it is necessary to make  $R_3$  very large compared with  $R_2$ .

Here we see that it is never necessary to connect two or more than two add gates or multiply gates consecutively, because the same effect can be obtained by a single gate carrying all the inputs. Hence we may assume that add gates and multiply gates alternate in multi-stage rectifier circuits as in the above example. Under this condition, it is necessary to increase the size of the gate resistor by a large factor for each stage in order to maintain the distinction of the levels 0 and 1. This requirement is so severe from the engineering viewpoint that most rectifier circuits practically constructed do not have more than two stages.

As a consequence of confining the number of stages within two, the problem of synthesizing minimal rectifier circuits, *i.e.*, circuits with the minimal number of rectifiers is reduced to the problem of finding minimal normal formulas of Boolean functions, because any function which is neither a normal sum nor a normal product requires a circuit with more than two stages. When we use a normal product (product), the number of rectifiers needed is the sum of the number of total occurrences of literals and the number of clauses (factors). But, when single literals appear in the minimal sum (product), their number must be subtracted from the above sum, because no rectifier is needed for producing the voltage represented by a single literal. The number of rectifier thus determined may be denoted by  $R_s$  and  $R_p$  respectively for the minimal sum and the minimal product. Then, a minimal circuit can be synthesized by using the minimal formula giving the smaller one between  $R_s$  and  $R_p$ .

*Example 6.2.1.* Synthesize a minimal rectifier circuit realizing the function:

$$f(w, x, y, z) = \sum(4, 7, 8, 11, 12, 15).$$

*Solution.* This function has the minimal sum:

$$f = wy'z' + wyz + xy'z' + xyz,$$

and the minimal product:

$$f = (w + x)(y' + z)(y + z').$$

The  $R_s$  and  $R_p$  are now obtained as  $R_s = 16$  and  $R_p = 9$ . Therefore the minimal product is preferred, and we obtain the minimal circuit:

$$B_3[F_2(w, x), F_2(y', z), F_2(y, z')].$$

*Example 6.2.2.* Synthesize a minimal rectifier circuit realizing the function:

$$f(w, x, y, z) = \sum(0, 3, 7, 11, 12, 13, 14, 15).$$

*Solution.* This function has the minimal sum:

$$f = wx + yz + w'x'y'z',$$

and the minimal product:

$$f = (w' + x + y)(w + x' + z)(w + y + z')(x + y' + z).$$

It follows that  $R_s = 11$  and  $R_p = 16$ . Therefore the minimal sum is preferred, and we obtain the minimal circuit:

$$F_3[B_2(w, x), B_2(y, z), B_4(w', x', y', z')].$$

*Example 6.2.3.* Synthesize a minimal rectifier circuit realizing the function:

$$f = \sum(5, 6, 8, 11, 12, 13, 14, 15)$$

*Solution.* This function is self-dual and has the minimal sum:

$$f = wy'z' + wyz + xy'z + xyz',$$

and the minimal product:

$$f = (w + y' + z')(w + y + z)(x + y' + z)(x + y + z').$$

It follows that  $R_s = R_p = 16$ . Therefore there are two minimal circuits each requiring 16 rectifiers. They are given by

$$F_4[B_3(w, y', z'), B_3(w, y, z), B_3(x, y', z), B_3(x, y, z')]$$

$$\text{and } B_4[F_3(w, y', z'), F_3(w, y, z), F_3(x, y', z), F_3(x, y, z')].$$

### 6.3. Vacuum Tube Switching Circuits

In this section, we deal with vacuum tube switching circuits which are formed from triodes and pentodes. Vacuum tube circuits containing diodes or any kind of rectifiers will not be considered here.

When a vacuum tube is used in a switching circuit, it is generally so operated that, when the high input voltage represented by 1 is applied to the grid or grids, the plate current takes the saturated value, and, when the low input voltage represented by 0 is applied to the grid or grids, the plate current is cut off. In this mode of operation, the plate current is almost insensitive to small fluctuations of the input voltages. Unlike the rectifier circuit, it is not necessary to restrict the vacuum tube circuit within two stages by virtue of amplifying action of vacuum tubes. The complement gate is readily constructed with a single triode as shown in Fig. 6.3.1. The plate is connected to the high supply voltage through a plate resistor  $R_p$  and to the low supply voltage through a potentiometer  $R$ . The input  $e$  is applied to the grid and the output  $e_0$  is taken from an appropriate point of the potentiometer.

When  $e$  is at its high level, the tube conducts and the plate voltage drops. When  $e$  is at its low level, the plate current is cut off and the plate voltage becomes the high supply voltage. Thus the variation of the grid voltage  $e$  induces the change of the plate voltage in the opposite direction. Therefore the output voltage  $e_0$  is the complement  $e'$  of the input voltage  $e$ . The potentiometer  $R$  is

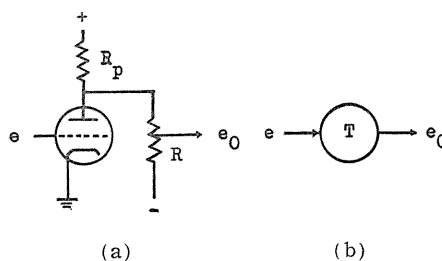


FIG. 6.3.1

to adjust the levels of the output voltage relative to the ground. This gate is represented by the symbolical diagram shown in (b).

The twin-triode gate is shown in Fig. 6.3.2. The two triodes are connected with a common plate resistor  $R_p$  which has a very large resistance compared with the tube plate resistance.

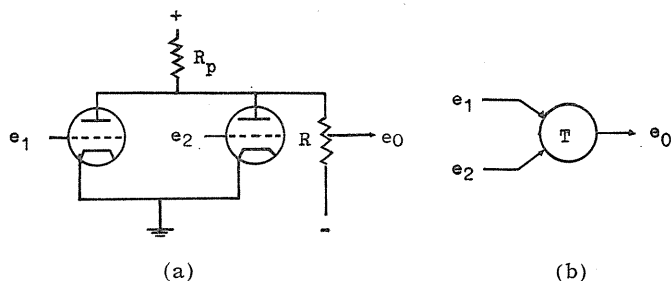


FIG. 6.3.2

When the high input voltage is applied to one of the triodes, the corresponding triode conducts and the plate voltage drops. Only when both inputs are low, the plate voltage is high. The relation between the inputs and the output is given by the following table, which indicates that  $e_0 = e'_1 e'_2$ . Note that the gate performs the Peirce's operation on the inputs. This gate can be extended to include any number of triods. If  $n$  triodes are connected with a common plate resistor, the output is given by  $e_0 = e'_1 e'_2 \cdots e'_n$ .

TABLE 6.3.1

$e_1$	$e_2$	$e_0$
0	0	1
0	1	0
1	0	0
1	1	0

The pentode gate is shown in Fig. 6.3.3. In this gate, the plate current flows only when both grid voltages are high. The relation between the inputs and the output is given by Table 6.3.2, which indicates that  $e_0 = e'_1 + e'_2$ . Note that the gate performs the Sheffer's operation on the inputs. Unlike the triode gate, this relation cannot practically be extended to include more than two input variables, because there is no vacuum tube with more than two grids which conducts only if all the grid voltages are high.

The common plate resistor connection may be applied to any number of triodes and pentodes. When any number of vacuum tubes are connected with

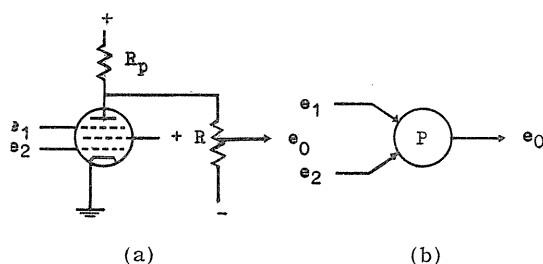


FIG. 6.3.3

TABLE 6.3.2

$e_1$	$e_2$	$e_0$
0	0	1
0	1	1
1	0	1
1	1	0

a common plate resistor, the plate voltage drops and the output voltage becomes low, if at least one of the tubes conducts. Accordingly, the output is the product of the outputs of the component tubes. This property of the common plate resistor connection is very useful in the synthesis of vacuum tube circuits, because the product of any number of functions can be realized by this connection without any additional tubes. For example, when a complement gate and a pentode gate are operated with a common plate resistor as shown in Fig. 6.3.4, the output is given by  $e_0 = e'_1(e'_1 + e'_3)$ .

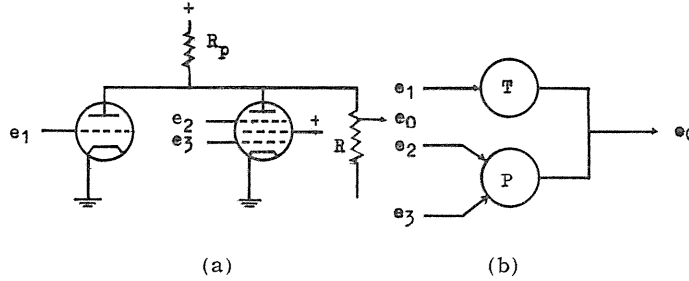


FIG. 6.3.4

When two triodes are operated with a common cathode resistor as shown in Fig. 6.3.5, the output is high if and only if at least one of the input is high. Thus, the output is given by  $e_0 = e_1 + e_2$ . This gate is called the cathode follower gate. Any number of triodes can be combined into a cathode follower gate. The switching action of a cathode follower gate including  $n$  triodes is given by  $e_0 = e_1 + e_2 + \dots + e_n$ .

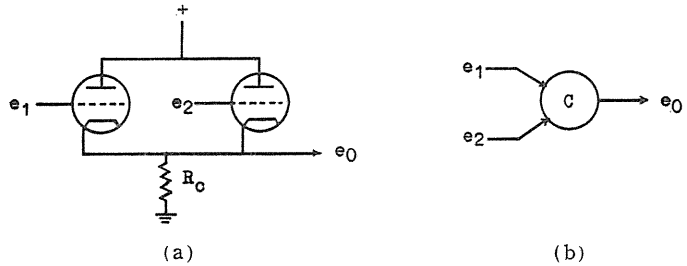


FIG. 6.3.5

Now we introduce the following four symbols.

$$P_n(e_1, e_2, \dots, e_n) = e'_1 e'_2 \dots e'_n.$$

$$S_n(e_1, e_2, \dots, e_n) = e'_1 + e'_2 + \dots + e'_n.$$

$$F_n(e_1, e_2, \dots, e_n) = e_1 + e_2 + \dots + e_n.$$

$$B_n(e_1, e_2, \dots, e_n) = e_1 e_2 \dots e_n.$$

Thus,  $P_1(e)$  represents a complement gate with the input  $e$ ,  $P_n(e_1, e_2, \dots, e_n)$  a triode gate with the inputs  $e_1, e_2, \dots, e_n$ ,  $S_2(e_1, e_2)$  a pentode gate with the inputs

$e_1$  and  $e_2$ , and,  $F_n(e_1, e_2, \dots, e_n)$  a cathode follower gate with the inputs  $e_1, e_2, \dots, e_n$ . There is no vacuum tube gate represented by  $S_n$  for  $n > 2$  and  $B_n$ . Hence these symbols are useless in vacuum tube circuits, but they are useful in transistor circuits as will be seen in the next section. Note that  $P$ ,  $S$  and  $F$  stand for "Peirce", "Sheffer" and "follower" respectively, and further that  $F$  and  $B$  have the same meaning as in the preceding section. With these symbols and with the convention that the common plate resistor connection of any number of gates is represented by the product of the corresponding symbols, and vacuum tube circuit can be represented by a symbolical expression. For example, the circuit of Fig. 6.3.6 is represented by  $S_2[F_2(u, v), P_1(w)]P_2(x, y)$ , and that of Fig. 6.3.7 by  $S_2(x, p)S_2[x', P_1(p)], p = P_2(y, z)$ .

When we use the symbolical representation, we must pay our attention to the fact that  $F$  may be used as an argument of, but may not be multiplied by any other symbol, because it is impossible to operate a cathode follower gate and any other gate with a common plate resistor. Thus, for example  $F_2(x, y)S_2(x, y)$  is meaningless.

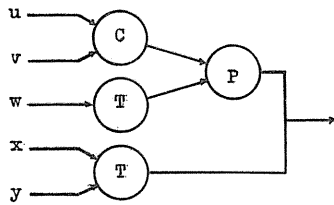


FIG. 6.3.6

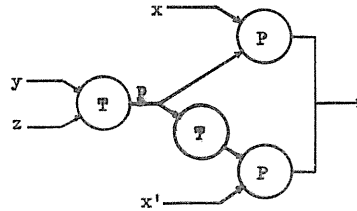


FIG. 6.3.7

Any Boolean function can be realized by twin-triode gates or pentode gates alone, because a twin-triode gate performs Peirce's operation and a pentode gate performs Sheffer's operation. However, in order to synthesize simple circuits, it is necessary to use all kinds of gates combined appropriately. A common standard of simplicity of vacuum tube circuits formed from triodes and pentodes is to count the number of control grids assuming that all the input variables and their complements are available as the input voltages. When we represent a vacuum tube circuit by a symbolical expression, the number of control grids is given by the sum of suffices of all the symbols appearing in the expression. Thus, for example, the circuit represented by the expressions

$$S_2(w, p)P_1[P_2(w, p)S_2(y', z')], p = S_2(x, y)S_2(x', y')$$

requires 11 grids.

A general stratagem for the synthesis of vacuum tube circuits is to try to use as many common plate resistor connections as possible. Consequently, we must pay more attention to the minimal product than to the minimal sum, or in other words, we must pay more attention to the minimal sum of the complement of the given function than the minimal sum of the function itself. The above paraphrase will be justified as follows: Not only the minimal product of a function is usually obtained from the minimal sum of its complement, but also the

latter is actually more convenient than the former in the synthesis of vacuum tube circuits. For example, let us consider the function  $f = (w + x')(y + z')$ . This function is realized by the circuit  $S_2(w', x)S_2(y', z)$ . When we start with the minimal product  $f = (w + x')(y + z')$ , we must turn every literal into its complement before entering it as an argument of  $S_2$  symbols. When we start with the minimal sum  $f' = w'x + y'z$ , on the contrary, such complementations are not required and every literal can be entered directly as an argument of  $S_2$  symbols.

Another stratagem is to try to find a good substitution. Accordingly, the preliminary examination of the functional separability is indispensable. Furthermore, since the usefulness of substitutions is not restricted to the case of functionally separable functions, it is important to grasp particular structure of functions by some means, for instance, by Karnaugh maps.

Now, we shall work out several examples for illustrating some typical situations encountered in the synthesis of vacuum tube circuits.

*Example 6.3.1.* Synthesize a circuit for the function  $f = x \oplus y$ .

*Solution.* Since  $f' = x'y' + xy$ ,  $f$  can be realized by the circuit:

$$S_2(x, y)S_2(x', y').$$

This circuit requires 4 grids and is minimal. Another circuit for this function is

$$S_2(x, y)P_1[P_2(x, y)].$$

This circuit requires 5 grids and is not minimal. But, if additional complement gates are necessary to produce  $x'$  and  $y'$ , the circuit is simpler than the previous one. Moreover, if  $x$  and  $y$  are not the input variables but the functions  $p$  and  $q$  of the input variables, complement gates are necessary to produce  $p'$  and  $q'$ , and therefore, the latter is simpler than the former. For example, consider the function  $f = w \oplus x \oplus y \oplus z$ . Using substitutions

$$p = w \oplus x = S_2(w, x)S_2(w', x')$$

and

$$q = y \oplus z = S_2(y, z)S_2(y', z'),$$

$f$  can be realized by the circuit:

$$S_2(p, q)P_1[P_2(p, q)].$$

This circuit requires 13 grids and is probably minimal.

Further, this function illustrates that the quality of the circuit depends upon the ways of substitutions. If we use the substitutions

$$r = w \oplus x = S_2(w, x)S_2(w', x')$$

and

$$s = y \oplus z = S_2(y, z)P_1[P_2(r, y)]$$

we obtain the circuit:

$$S_2(r, z)P_1[P_2(r, z)].$$

This circuit requires 14 grids and is less simple than the previous one.

*Example 6.3.2.* Synthesize a circuit for the function:

$$f = w'x' + w'y + x'y'z.$$

*Solution.* The minimal sum of  $f'$  is given by

$$f' = wx + wy + wz' + xy.$$

Thus, we obtain the circuit

$$S_2(w, x) S_2(w, y) S_2(w, z') S_2(x, y)$$

requiring 8 grids. But, if we simplify  $f'$  as

$$f' = w(x + y + z') + xy,$$

we obtain the circuit

$$S_2[w, F_3(x, y, z')] S_2(x, y)$$

requiring only 7 grids. This example illustrates that the factoring is sometimes profitable.

*Example 6.3.3.* Synthesize a circuit for the function:

$$f = w'y + w'z + x'y' + x'z'.$$

*Solution.* The minimal sum of  $f'$  is given by

$$f' = wx + wyz + x'y'z'.$$

Using this form directly, we obtain the circuit

$$S_2(w, x) P_2[P_3(w', y', z'), P_3(x', y, z)]$$

requiring 10 grids. Further, when we rewrite  $f'$  as

$$f' = w[x + (y' + z')'] + xy'z',$$

we obtain another circuit

$$S_2[w, S_2[x', S_2(y, z)]] P_1[P_3(x', y, z)]$$

requiring also 10 grids. No better circuit can be obtained if we start with the minimal sum of  $f'$ . But, when we start with the form

$$f = w'(y + z) + x'(y' + z')$$

and use a cathode follower gate, we obtain the circuit

$$F_2[P_1(w) S_2(y', z'), P_1(x) S_2(y, z)]$$

requiring only 8 grids. Thus, it is not always profitable to adhere to the use of common plate resistor connections. Sometimes better circuits can be synthesized from the minimal sum of the given function by means of a cathode follower gate.

*Example 6.3.4.* Synthesize a circuit for the function:

$$f = wy' + xz' + yz.$$



*Solution.* The minimal sum of  $f'$  is given by

$$f' = w'x'y' + w'y'z + x'yz'.$$

Rewriting this as

$$f' = w'y'(x' + z) + x'yz',$$

we obtain the circuit  $P_2[P_2(w, y)S_2(x, z'), P_3(x, y', z)]$  requiring 9 grids. On the other hand, since it is seen from the minimal sum of  $f$  that  $f'$  can be realized by the circuit

$$S_2(w, y')S_2(x, z')S_2(y, z),$$

we obtain another circuit

$$P_1[S_2(w, y')S_2(x, z')S_2(y, z)]$$

requiring only 7 grids. As shown by this example, it is sometimes profitable to utilize complementary circuits. It should be noted that an additional complement gate is not always required. When the complementary circuit is of the form  $F_n(\dots)$  or  $P_n(\dots)$ , no additional complement gate is required and the desired circuit is given by  $P_n(\dots)$  or by  $F_n(\dots)$  respectively.

*Example 6.3.5.* Synthesize a circuit for the function:

$$f = wx + wy + w'x'y'z.$$

*Solution.* This function is functionally separable and can be written as

$$f = wp' + w'pz,$$

where  $p = x'y'$ . The minimal sum of  $f'$  is given by

$$f' = w'(p' + z') + wp.$$

Therefore we obtain the circuit

$$S_2(w, p)S_2[w', S_2(z, p)], p = P_2(x, y).$$

The circuit requires 8 grids and is probably minimal.

*Example 6.3.6.* Synthesize a circuit for the function:

$$f = wxy' + wx'y + wx'z' + w'xy + w'x'y'z.$$

*Solution.* This function is functionally separable and can be written as

$$f = wp' + w'p = (w' + p')(w + p),$$

where

$$p = (x' + y)(x + y')(x + z).$$

Therefore we obtain the circuit

$$S_2(w, p)P_1[P_2(w, p)], p = S_2(x, y')S_2(x', y)S_2(x', z')$$

requiring 11 grids. The structure like this may be recognized by means of Karnaugh maps as follows. The map of the function of this example is shown

in Fig. 6.3.8. Now, if we fold up the  $w'$ -region onto the  $w$ -region, it will be observed that 1's coming from the  $w'$ -region fill up exactly the vacant cells of the  $w$ -region. This means that the  $w$ -residue and the  $w'$ -residue are the complement of one another.

When we expand a function with respect to  $x$ ,  $y$  or  $z$  in stead of  $w$ , we may fold the map as shown in Fig. 6.3.9.

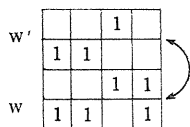


FIG. 6.3.8

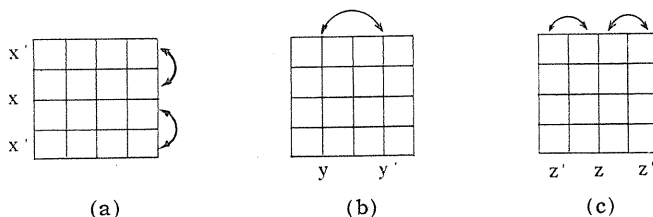


FIG. 6.3.9

The folding procedure is also applied to the cases where the function has not exactly the same structure as above. It is often successful in finding good substitutions.

*Example 6.3.7.* Synthesize a circuit for the function:

$$f = wxy' + wxz + wx'y'z' + w'x'yz.$$

*Solution.* The function is plotted in a Karnaugh map as shown in Fig. 6.3.10. Now, if we fold up the  $x'$ -region onto the  $x$ -region, it will be found that 1's of the  $x'$ -region go to vacant cells of the  $x$ -region but do not fill up all of them. This implies that the  $x$ -residue is included in the complement of the  $x'$ -residue and vice versa. Let the  $x'$ -residue and the  $x$ -residue be denoted by  $p$  and  $q$  respectively. Then there are a function  $g$  and another function  $h$  such that  $q = gb'$  and  $p = hq'$ . Since  $g$  and  $h$  are not determined uniquely, it is desirable to find the simplest ones. This can be done as follows. First,

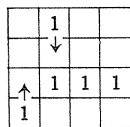


FIG. 6.3.10

we enter marks in the vacant cells of the  $x$ -region, each of which is filled with a 1 coming from the  $x'$ -region. Then we select some of marked cells so that the selected cells, when combined with the original 1-cells, may form the simplest function in the  $x$ -region. As shown in (a) of Fig. 6.3.11, the best

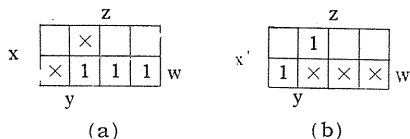


FIG. 6.3.11

choice in the present case is to select the marked cell in the bottom row. Therefore, the simplest form of  $g$  is  $g = w$ . The similar procedure, when applied to the  $x'$ -region, reveals that the simplest form of  $h$  is  $h = y(w + z)$ . Thus, we obtain the relations  $q = wp'$  and  $p = y(w + z)q'$ . Using the first relation,

$f$  can be rewritten as

$$f = wxp' + x'p = (x' + p')(wx + p) = (x' + p')[(w' + x')p']',$$

where  $p = y(w' + z')(w + z)$ .

Hence,  $f$  can be realized by the circuit

$$S_2(x, p) P_1[P_1(p) S_2(w, x)], p = P_1(y') S_2(w, z) S_2(w', z').$$

This circuit requires 11 grids and is perhaps minimal.

The present method is generally more successful when not the given function itself but its complement has the structure of this example. For example, the circuit realizing  $f'$  is immediately synthesized from  $f = wxp' + x'p$  as

$$S_2(x', p) P_1[P_1(w', x', p)].$$

*Example 6.3.8.* Synthesize a circuit for the function:

$$f = w'y + wx'y' + w'xz' + x'y'z.$$

*Solution.* The Karnaugh map of this function is shown in Fig. 6.3.12. Now imagine that the cell at the right upper corner contains a 1, then map will represent a function which can be decomposed into two terms  $x'y'$  and  $w'(y + z')$ . Since any of the 1-cell of  $f$  is included in exactly one of and the imaginary 1-cell in both of the two terms, we conclude that  $f$  can be expressed as

$$f = x'y' \oplus w'(y + z').$$

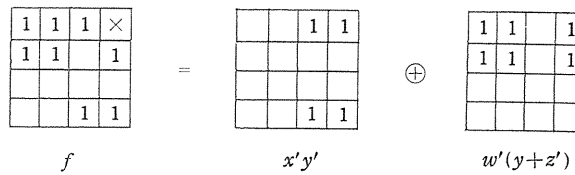


FIG. 6.3.12

Therefore, using the substitutions

$$p = x'y' = P_2(x, y)$$

and

$$q = w'(y + z') = P_1(w) S_2(y', z'),$$

we obtain the circuit

$$S_2(p, q) P_1[P_1(p, q)].$$

This circuit requires 10 grids and is probably minimal.

As illustrated by this example, it is sometimes of advantage to assume the structure  $f = g \oplus h$ . One method to find simple  $g$  and  $h$  functions is to set up some imaginary 1-cells in the map of  $f$  and see whether the resulting map may represent a sum of two simple functions such that any of the real 1-cells is included in exactly one of and any of the imaginary 1-cells in both of them.

*Example 6.3.9.* Synthesize a circuit for the function:

$$f = S_{1,2,4}(w, x, y, z).$$

*Solution.* Symmetric functions are the most difficult to handle in the synthesis of vacuum tube circuits. One way to solve the present problem is as follows. We write  $f' = S_{0,3}(w, x, y, z)$  in the form:

$$f' = (w' + x')(w + x)yz + wx(y' + z')(y + z) + w'x'y'z'.$$

Then, putting  $p = wx$ ,  $q = w'x'$ ,  $r = yz$ ,  $s = y'z'$ , we obtain

$$f' = p'q'r + pr's' + qs.$$

Hence,  $f$  can be realized by the circuit

$$S_2(q, s)S_2[p, P_2(r, s)]S_2[r, P_2(p, q)],$$

where  $p = P_2(w', x')$ ,  $q = P_2(w, x)$ ,  $r = P_2(y', z')$ ,  $s = P_2(y, z)$ . This circuit requires 18 grids.

Another way is to apply the method of Example 6.3.8. It will be found from the Karnaugh map that  $f'$  can be expressed as

$$f' = g \oplus h$$

where

$$g = w'x' + yz$$

and

$$h = (w' + x)(w + x')(y + z),$$

as shown in Fig. 6.3.13. Thus  $f$  can be written as

$$f = p \oplus q$$

where

$$p = g' = (w + x)(y' + z')$$

and

$$q = h = (w' + x)(w + x')(y + z).$$

Hence, we obtain the circuit

$$S_2(p, q)P_1[P_2(p, q)]$$

where  $p = S_2(w', x')S_2(y, z)$  and  $q = S_2(w, x')S_2(w', x)S_2(y', z')$ .

This circuit requires only 15 grids and is far simpler than the previous one.

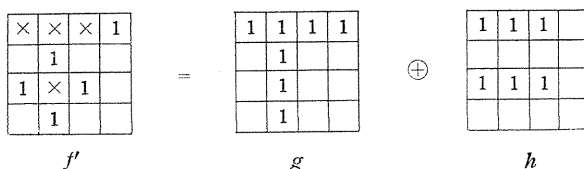


FIG. 6.3.13

#### 6.4. Transistor Switching Circuits

The transistor is a very promising component of electronic circuits. It is made very compact, it requires nothing analogous to the incandescent filament of the vacuum tube and it operates with a very small total power drain. Owing

to these advantages over the vacuum tube, the transistor is beginning to displace the vacuum tube in many fields.

In electronic switching circuits, various types of transistors are used. But here we shall confine the discussions to the PNP and NPN transistors only. These two types of transistors are usually represented by the symbolic diagrams as shown in Fig. 6.4.1. Three electrodes are called *emitter*, *collector* and *base*. The emitter-base circuit and the collector-base circuit are equivalent to rectifiers, and their forward directions are from  $E$  to  $B$  and from  $C$  to  $B$  in a PNP transistor, and from  $B$  to  $E$  and from  $B$  to  $C$  in an NPN transistor.

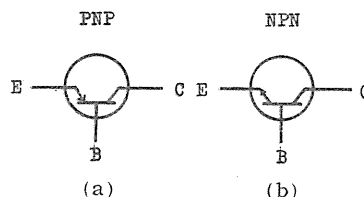


FIG. 6.4.1

The proper mode of operation of a transistor is to control the backward current of the  $C$ - $B$  circuit by the forward current of the  $E$ - $B$  circuit. Accordingly, the polarities of the emitter and the collector relative to the base are positive (negative) and negative (positive) respectively in a PNP (an NPN) transistor. These transistors are commonly operated by grounding the emitter, applying the input to the base and taking the output from the collector. When operated in this way, a transistor bears close analogy with a triode. In the analogy, the emitter, the base and the collector of a transistor correspond to the cathode, the grid and the plate of a triode respectively. Especially, between an NPN transistor and a triode, the polarities of the corresponding electrodes are the same, and therefore, they are almost equivalent as switching elements. On the other hand, between a PNP transistor and a triode, the polarities of the corresponding electrodes are opposite, and therefore, they are almost dual to one another. Of course, a PNP transistor and an NPN transistor are perfectly dual to one another.

The complement gate can be formed from a single PNP transistor or a single NPN transistor as shown in Fig. 6.4.2.

When the base voltage  $e$  is high (low) in the PNP (NPN) gate, no  $e$ -emitter current flows, and collector current is cut off. In this state, the output voltage  $e_0$  is equal to the supply voltage, *i.e.*, it is low (high). When  $e$  is low (high), on the contrary, the emitter current flows and induces the collector current. In this state, the output voltage becomes high (low) by the presence of the collector resistor  $R_c$ . Thus, it has been shown that each of these gates is really a complement gate.

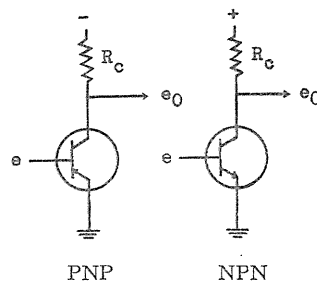


FIG. 6.4.2

When two transistors are operated with a common collector resistor as shown in Fig. 6.4.3, the output voltage is equal to the supply voltage only if both transistors are turned off. Hence the relation between the inputs and the output is given by Table 6.4.1, which indicates that  $e_0 = e'_1 + e'_2$  in the PNP gate and  $e_0 = e'_1 e'_2$  in the NPN gate.

Each of these gates is called the *parallel gate*. Any number of transistors

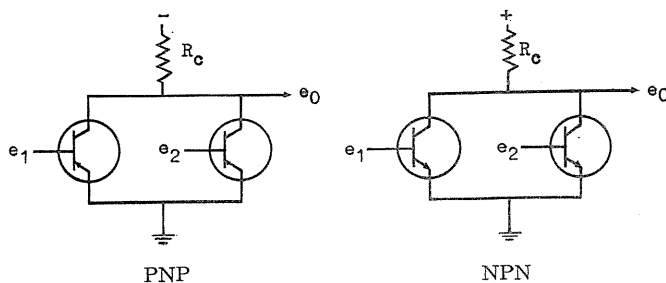


FIG. 6.4.3

TABLE 6.4.1

$e_1$	$e_2$	$e_0$	$e_1$	$e_2$	$e_0$
0	0	1	0	0	1
0	1	0	0	1	1
1	0	0	1	0	1
1	1	0	1	1	0
PNP			NPN		

can be combined into a parallel gate. The output of the parallel gate including  $n$  PNP (NPN) transistors is given by

$$e_0 = e'_1 + e'_2 + \cdots + e'_n (e_0 = e'_1 e'_2 \cdots e'_n).$$

The transistor gate corresponding to the pentode gate is a series connection of two transistors. This connection is very rare in vacuum tube circuits but it is rather common

in transistor circuits. In each of the gates shown in Fig. 6.4.4, the current flows through  $R$  and hence the output voltage is different from the supply voltage, only when both transistors are conducting. Therefore we obtain Table 6.4.2 indicating  $e_0 = e'_1 e'_2$  in the PNP gate and  $e_0 = e'_1 + e'_2$  in the NPN gate. Each of these gates is called the *series gate*. Unlike the pentode gate, the series gate can be extended to include any number of inputs by connecting transistors, each carrying a single input, in series. The output of the series gate consisting of  $n$  PNP (NPN) transistors is given by

$$e_0 = e'_1 e'_2 \cdots e'_n (e_0 = e'_1 + e'_2 + \cdots + e'_n).$$

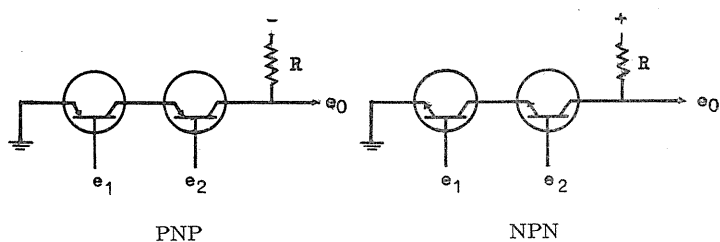


FIG. 6.4.4

TABLE 6.4.2.

$e_1$	$e_2$	$e_0$	$e_1$	$e_2$	$e_0$
0	0	1	0	0	1
0	1	0	0	1	1
1	0	0	1	0	1
1	1	0	1	1	0
PNP			NPN		

Now we have three gates for each type of transistor. As in the case of vacuum tube gates, any number of these gates formed from the same type of transistor can be operated with a common collector resistor. It will be easily observed that the output of a common collector resistor connection of PNP (NPN) gates is the sum (product) of

the outputs of component gates. Thus, for example, the outputs of the circuits shown in Fig. 6.4.5 are  $e'_1 + e'_2 e'_3$  and  $e'_1(e'_2 + e'_3)$  respectively.

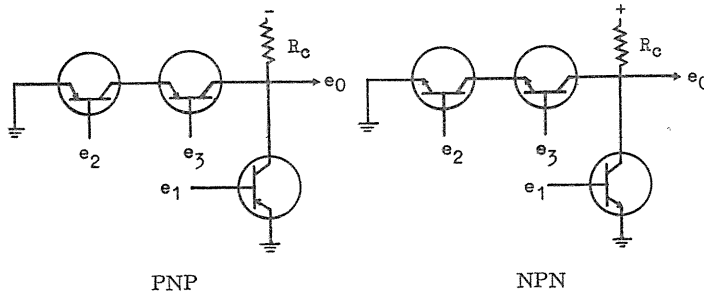


FIG. 6.4.5

The gate which is analogous to the cathode follower gate is the *emitter follower gate*. Fig. 6.4.6 shows the emitter follower gates formed from two transistors.

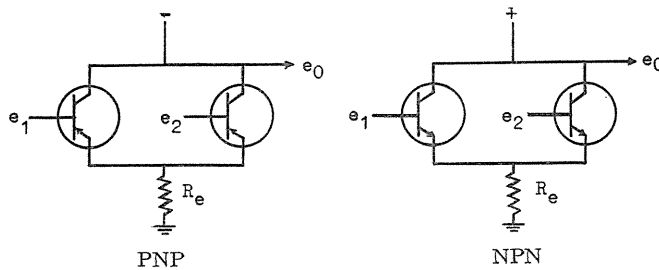


FIG. 6.4.6

In these gates, two transistors are connected with a common emitter resistor. Since their outputs are equal to the emitter voltage only when both transistors are turned off, we obtain Table 6.4.3 which indicates that the output of the PNP gate is  $e_0 = e_1 e_2$  and that of the NPN gate is  $e_0 = e_1 + e_2$ .

The emitter follower gate may include any number of transistors. The output of the emitter follower gate including  $n$  PNP (NPN) transistors is given by

$$e_0 = e_1 e_2 \cdots e_n (e_0 = e_1 + e_2 + \cdots + e_n).$$

TABLE 6.4.3

$e_1$	$e_2$	$e_0$	$e_1$	$e_2$	$e_0$
0	0	0	0	0	0
0	1	0	0	1	1
1	0	0	1	0	1
1	1	1	1	1	1

PNP

NPN

All the transistor gates have now been introduced. Any transistor switching circuit can be synthesized by means of these gates. As in the case of vacuum tube circuits, it is convenient to represent transistor circuits by symbolical expressions. For this purpose, we shall employ the same symbols introduced in the

TABLE 6.4.4

Symbol	Switching Action	PNP Gate	NPN Gate
$P_1(e)$	$e'$	—	Complement Gate
$S_1(e)$	$e'$	Complement Gate	—
$P_n(e_1, e_2, \dots, e_n)$	$e'_1 e'_2 \cdots e'_n$	Series Gate	Parallel Gate
$S_n(e_1, e_2, \dots, e_n)$	$e'_1 + e'_2 + \cdots + e'_n$	Parallel Gate	Series Gate
$F_n(e_1, e_2, \dots, e_n)$	$e_1 + e_2 + \cdots + e_n$	—	Emitter Follower Gate
$B_n(e_1, e_2, \dots, e_n)$	$e_1 e_2 \cdots e_n$	Emitter Follower Gate	—

preceding section. The meaning of the symbols is given in the above table.

Further, it is agreed that a sum (product) of any number of  $P$  symbols and  $S$  symbols represents a common collector resistor connection of the corresponding PNP (NPN) gates. It should be noted that any sum or product including  $F$  symbols or  $B$  symbols is meaningless. Thus, for example,  $P_2(e_1, e_2) + B_2(e_1, e_2)$ ,  $S_2(e_1, e_2)F_2(e_1, e_2)$  and  $F_2(e_1, e_2) + B_2(e_1, e_2)$  are all meaningless.

The most reasonable standard of simplicity of transistor switching circuits is to count the number of transistors required, assuming that all the input variables and their complements are available as the input voltages. When a circuit is represented by a symbolical expression, the number of transistors required is given by the sum of suffices of all the symbols appearing in the expression.

With the above standard of simplicity and the symbols, the synthesis of NPN transistor circuits is formally almost the same as that of vacuum tube circuits. Accordingly, all the methods for the synthesis of vacuum tube circuits are equally applicable to the synthesis of NPN transistor circuits. The only one difference is that the symbol  $S_n$  for  $n > 2$  can be realized by a single NPN transistor gate but not by any single vacuum tube gate.

On the other hand, the synthesis of PNP transistor circuits is perfectly dual of that of NPN transistor circuits. Thus, to any method of the synthesis of NPN transistor circuits, there corresponds a dual method of the synthesis of PNP transistor circuits, and vice versa. For example, the predominance of the minimal sum over the minimal product in the synthesis of PNP transistor circuits corresponds to the predominance of the minimal product over the minimal sum in the synthesis of NPN transistor circuits.

Now in the rest of this section, we shall work out some examples, laying special emphasis on the difference between transistor circuits and vacuum tube circuits and the duality between PNP and NPN transistor circuits.

*Example 6.4.1.* Synthesize transistor switching circuits for the function:

$$f = w' + xy' + x'y + x'z'.$$

*Solution.* The minimal sum of  $f'$  is given by

$$f' = wxy + wx'y'z.$$

From these minimal sums, we can immediately synthesize the PNP transistor circuit



$$S_1(w) + P_2(x', y) + P_2(x, y') + P_2(x, z),$$

and the NPN transistor circuit

$$S_3(w, x, y) S_4(w, x', y', z).$$

Both circuits require 7 transistors and are probably minimal.

*Example 6.4.2.* Synthesize transistor switching circuits for the function:

$$f = w'xy + wx'y' + w'y'z + x'yz'.$$

*Solution.* The minimal sum of  $f'$  is given by

$$f' = wx + w'y'z' + x'yz.$$

From these minimal sums, we can synthesize the PNP transistor circuit

$$P_3(w, x', y') + P_3(w', x, y) + P_3(w, y, z') + P_3(x, y', z),$$

and the NPN transistor circuit

$$S_2(w, x) S_3(w', y', z') S_3(x', y, z).$$

The latter circuit requires 8 transistors and is minimal, but the former circuit requires 12 transistors and is not minimal. A better circuit can be synthesized as follows. First, we synthesize a complementary circuit

$$P_2(w', x') + P_3(w, y, z) + P_3(x, y', z')$$

from the minimal sum of  $f'$ . Then complementing this circuit by a complement gate, we obtain

$$S_1[P_2(w', x') + P_3(w, y, z) + P_3(x, y', z')].$$

This circuit requires 9 transistors and is perhaps minimal.

Note that the complementary circuit can also be obtained from the NPN transistor circuit by replacing  $S$  symbols with  $P$  symbols, multiplications with additions and complementing all the literals. This procedure is always valid by virtue of the duality of PNP and NPN transistor circuits. In general, when we are given a PNP (an NPN) transistor circuit, the complementary NPN (PNP) circuit will be obtained by interchanging  $P$  symbols with  $S$  symbols,  $B$  symbols with  $F$  symbols, additions with multiplications, and complementing all the literals appearing in the symbolical expression.

*Example 6.4.3.* Synthesize transistor switching circuits for the function:

$$f = wx + wy + wz + xyz.$$

*Solution.* The minimal sum of  $f'$  is given by

$$f' = w'x' + w'y' + w'z' + x'y'z'.$$

This time, it is of advantage to rewrite these minimal sums as

$$f = w(x'y'z')' + xyz$$

and

$$f' = w'(x' + y' + z') + x'y'z'$$

respectively. From these expressions, we can synthesize the PNP transistor circuit

$$P_2[w', P_3(x, y, z)] + P_3(x', y', z')$$

and the NPN transistor circuit

$$S_2[w', S_3(x, y, z)]S_3(x', y', z').$$

Both circuits require 8 transistors and are perhaps minimal.

*Example 6.4.4.* Synthesize transistor switching circuits for the function:

$$f = w'xy' + wx'y' + wx'z + w'y'z + w'yz'.$$

*Solution.* The minimal sum of  $f'$  is given by

$$f' = wx + w'yz + wyz' + w'x'y'z'.$$

First, we synthesize the NPN transistor circuit

$$S_2(w, x)S_3(w', y, z)S_3(w, y, z')S_4(w', x', y', z')$$

form the minimal sum of  $f'$ . This circuit requires 12 transistors.

Another equally simple NPN transistor circuit can be obtained as follows. Rewriting  $f$ , we obtain

$$f = wx'(y' + z) + w'(y' + z')(x + y + z).$$

Then, using an emitter follower gate, we synthesize the circuit

$$F_2[P_2(w', x)S_2(y, z'), P_1(w)S_2(y, z)S_3(x', y', z')].$$

Next, we shall synthesize a PNP transistor circuit. First, we synthesize the complementray NPN transistor circuit

$$P_2[P_2(w', x)S_2(y, z'), P_1(w)S_2(y, z)S_3(x', y', z')].$$

Then, applying the general rule stated in Example 6.4.2, we obtain the circuit

$$S_2[S_2(w, x') + P_2(y', z), S_1(w') + P_2(y', z') + P_3(x, y, z)].$$

This circuit requires also 12 transistors.

*Example 6.4.5.* Synthesize transistor switching circuits for the function:

$$f = w'y' + w'z' + x'y'z' + x'yz + xy'z + xyz'.$$

*Solution.* The minimal sum of  $f'$  is given by

$$f' = xyz + wx'y'z + wx'yz' + wxy'z'.$$

Applying the method used in Example 6.3.6,  $f'$  can be rewritten as

$$f' = zp + wz'p'$$

where

$$p = xy + wx'y',$$

or as

$$f' = zq' + wz'q$$

where

$$q = p' = w'x' + x'y + xy'.$$

Suppose we are synthesizing vacuum tube circuits, then the above two forms yield the circuits

$$S_2(z, p) P_1[P_3(w', z, p)]$$

where

$$p = S_2(w', x') S_2(x', y) S_2(x, y'),$$

and

$$S_2[P_2(w', z), q] P_1[P_2(z', q)]$$

where

$$q = S_2(x, y) P_1[P_3(w', x, y)]$$

respectively. Since the former circuit requires 12 grids and the latter requires 13 grids, the first form is preferred for the vacuum tube circuits. But, for the NPN transistor circuits, on the contrary, the second form is preferred. In fact, we obtain the circuit

$$S_2(z, p) P_1[P_3(w', z, p)]$$

where

$$p = S_2(w', x') S_2(x', y) S_2(x, y')$$

from the first form, and the circuit

$$S_3(w, z', q) P_1[P_2(z', q)]$$

where

$$q = S_2(x, y) S_3(w, x', y')$$

from the second form. The former circuit requires 12 transistors but the latter requires 11 transistors.

Next, a PNP transistor circuit will be synthesized. From the two forms of  $f'$ , we obtain

$$f = zp' + z'(w' + p) = zp' + (z + wp')'$$

and

$$f = z'q' + z'w' + zq$$

respectively.

These form yield the circuit

$$P_2(z', p) + S_1[S_1(z') + P_2(w', p)]$$

where

$$p = P_2(x', y') + P_3(w', x, y),$$

and the circuit

$$P_2(z, q) + P_2(w, z) + S_1[S_2(z, p)]$$

where

$$q = P_2(w, x) + P_2(x, y') + P_2(x', y)$$

respectively. The former requires 11 transistors but the latter requires 13 transistors. Hence, this time, the first form is preferred.

*Example 6.4.6.* Synthesize transistor switching circuits for the function:

$$f = w'xy' + wxy + wxz + wyz + w'x'y'z' + wx'y'z'.$$

*Solution.* Using the method explained in Example 6.3.8, it will be found from the Karnaugh map that  $f$  can be rewritten as

$$f = w(y + z') \oplus (xy' + x'yz'),$$

or

$$f = x'(y' + z') \oplus (wy + w'y' + y'z).$$

These two forms are equally good from the viewpoint of vacuum tube circuits, because each of them yields a circuit requiring 14 grids. But, from the viewpoint of transistor circuit, they are not equally good. That is: The first form is better for the PNP transistor circuit, and the second form is better for the NPN transistor circuit. The reason is as follows: First we synthesize NPN transistor circuits from the two forms. Then, we obtain the circuit

$$S_2(p, q) P_1[P_2(p, q)]$$

$$\text{where } p = P_1(w') S_2(y', z) \quad \text{and} \quad q = S_2(x, y) S_2(x', y') S_2(x', z)$$

from the first form, and the circuit

$$S_2(r, s) P_1[P_2(p, q)]$$

$$\text{where } r = P_1(x) S_2(y, z) \quad \text{and} \quad s = S_2(w', y) S_3(w, y', z')$$

from the second form. The former requires 14 transistors but the latter requires 13 transistors.

Next, we synthesize PNP transistor circuits. We obtain the circuit

$$P_2(p, q) + S_1[S_2(p, q)]$$

$$\text{where } p = S_1(w) + P_2(y, z') \quad \text{and} \quad q = P_2(x', y) + P_3(x, y', z)$$

from the first form, and the circuit

$$P_2(r, s) + S_1[S_2(r, s)]$$

$$\text{where } r = S_1(x') + P_2(y', z') \quad \text{and} \quad s = P_2(w', y') + P_2(w, y) + P_2(y, z')$$

from the second form. The former requires 13 transistors but the latter requires 14 transistors.

## 7. Minimal Switching Circuits for Boolean Functions of Four Variables

There is no general method for synthesizing minimal switching circuits of any kind under any standard of simplicity. Consequently, the only way to find minimal circuits off hand is to prepare a table of minimal circuits for all Boolean functions. In reality, it is sufficient to prepare a table of minimal circuits for all representatives of the type, because Boolean functions of the same type are

realized by minimal circuits which are substantially identical or can be so regarded under some conditions. The plan is feasible in the case of four variables, but, beyond this, it becomes extremely expensive.

The author endeavored to construct a complete table of minimal switching circuits for Boolean functions of four variables. The result is given in Appendix 2 under the title "Table of Minimal Switching Circuits for Boolean Functions of Four Variables". The table contains relay circuits, rectifier circuits, vacuum tube circuits and transistor circuits which are minimal or presumably minimal under the respective standards of simplicity. Since much time has been spent for the construction of the table, it may be permissible to assume that the table is free from errors.

### 7.1. Minimal Relay Switching Circuits

For each type of Boolean functions of four variables, one relay circuit which is minimal both in contacts and springs is given. But, when no such circuit exists, two circuits are given. Of course, one is minimal in contacts and the other is minimal in springs. It has been revealed that, *except five types, every type has a circuit which is minimal both in contacts and springs.*

When a series-parallel circuit and a bridge are equally simple, the series-parallel circuit is chosen. Series-parallel circuits are not given by circuit diagrams but algebraic expressions. Further, when a planar circuit and a nonplanar circuit are equally simple, the planar circuit is chosen. It is interesting to note that *planar bridges occupy a large proportion (about 66%) of all minimal circuits.*

The number of contacts and the number of springs are given in the columns "C" and "S" respectively. The table indicates that the *real upper bound of the number of contacts is not 14 (the Shannon's upper bound),<sup>1)</sup> but is 13.* Moreover, there is only two types (112 and 112') requiring 13 contacts. All other types require at most 12 contacts.

One more remarkable fact exists. That is: *With only one exception (69 and 69'), the minimal number of contacts are identical between complementary types.*

During the preparation of the table, the author became aware of the similar table of minimal relay circuits compiled by E. F. Moore. Moore's table was published as the Appendix to the book, "Logical Design of Electrical Circuits" by R. A. Higonnet and R. A. Grea. The author compared his initial table with Moore's table and found that some revisions are necessary for both tables. Some circuits adopted in the author's table were not minimal in contacts and some others were not minimal in springs. These errors were corrected in the table published here. On the other hand, 35 circuits of Moore's table are pointed out not to be minimal in springs. The simpler circuits of the author's table are distinguished by attaching the asterisk to the number of springs in the column "S". Incidentally, it may be added that Moore's table is based on the Harvard Table.

### 7.2. Minimal Rectifier Switching Circuits

For each type of Boolean functions of four variables, a minimal two-stage rectifier switching circuit is given. The circuit is given by the minimal sum in

<sup>1)</sup> See page 120.

the top line of each block. The number of rectifiers required for the circuit is shown in the column "R". If the number has an asterisk, the circuit is not minimal and another circuit must be synthesized on the basis of the minimal product. The minimal product can easily be obtained from the minimal sum given in the block for the complementary type. Evidently, the number of rectifiers required for the new circuit is given by the number of the column "R" for the complementary type. Surveying the table, it is seen that *the upper bound of the number of rectifiers is 40*.

### 7.3. Minimal Vacuum Tube Switching Circuits

For each type of Boolean function of four variables, a vacuum tube switching circuit with minimal number of control grids is given by the symbolical expression. Here it is assumed that no complement gate is necessary for producing the complements of the input variables. The number of grids is shown in the column "G".

By comparison with the Harvard Table,<sup>1)</sup> it has been found that *as many as 151 circuits of the present table are simpler than the corresponding circuits of the Harvard Table*. These circuits are distinguished by the asterisk attached to the number of grids in the column "G". For the sake of reference, the Harvard serial number of type is shown in the column "H".

As regards the upper bound of the number of grids, the table tells us that *it is not 20 (the value from the Harvard Table), but only 16*, and further, that there are only three types (58, 112, 176) requiring 16 grids.

### 7.4. Minimal Transistor Switching Circuits

For each type, of Boolean functions of four variables, a transistor switching circuit which is minimal in the number of transistors is given by the symbolical expression. Of course, it is assumed that no complement gate is needed for producing the complements of the input variables. The number of transistors is shown in the column "T". As a matter of fact, only NPN transistor circuits are given, since PNP transistor circuits will be easily obtained from NPN transistor circuits for the complementary types by interchanging *P* symbols with *S* symbols, *F* symbols with *B* symbols, by replacing multiplications by additions, and by complementing all the literals. Only for every type of functions of the dimension 8, both circuits are given in full. When a block contains only one symbolical expression, the expression represents the vacuum tube circuit and the NPN transistor circuit at the same time. It is seen that *about three quarters of the types have minimal vacuum tube circuits and transistor circuits with the same symbolical expressions*. For the remaining quarter of the types, transistor circuits give smaller numbers of transistors than the numbers of grids of the corresponding vacuum tube circuits. *The difference between these numbers is usually 1 and rarely 2, but never 3*.

As regards the upper bound of the number of transistors, the same thing is observed as in the case of vacuum tube circuits. That is: *Only three types (58, 112, 176) require 16 transistors, and all the others require at most 15 transistors*.

<sup>1)</sup> cf. Ref. 38.

**Appendix 1. Table of Boolean Functions of Four Variables**

Coordinates of Atoms

										1 2 3 4	
										1 1 1 2 2 2 3 3 3 4 4 4	
Atom	0	1	2	3	4	1	2	3	4	1	2
0	7	1	1	1	1	1	1	1	1	1	1
1	7	1	1	1	1	1	1	1	1	1	1
2	7	1	1	1	1	1	1	1	1	1	1
3	7	1	1	1	1	1	1	1	1	1	1
4	7	1	1	1	1	1	1	1	1	1	1
5	7	1	1	1	1	1	1	1	1	1	1
6	7	1	1	1	1	1	1	1	1	1	1
7	7	1	1	1	1	1	1	1	1	1	1
8	7	1	1	1	1	1	1	1	1	1	1
9	7	1	1	1	1	1	1	1	1	1	1
10	7	1	1	1	1	1	1	1	1	1	1
11	7	1	1	1	1	1	1	1	1	1	1
12	7	1	1	1	1	1	1	1	1	1	1
13	7	1	1	1	1	1	1	1	1	1	1
14	7	1	1	1	1	1	1	1	1	1	1
15	7	1	1	1	1	1	1	1	1	1	1

										1 2 3 4	
										1 1 1 2 2 2 3 3 3 4 4 4	
N	0	1	2	3	4	1	2	3	4	Standard Sum	Symmetry
1	8	0000	000000	0000	0						(1, 2, 3, 4), [1], [2], [3], [4]
2	7	1111	111111	1111	1	15					(1, 2, 3, 4)
3	6	2220	220200	2000	0	14	15				(1, 2, 3), [4]
4	6	2200	200002	0022	2	12	15				(12), (34), [34]
5	6	2000	000222	2220	0	8	15				(2, 3, 4), [234]
6	6	0000	222222	0000	2	0	15				(1, 2, 3, 4), [1234]
7	5	3311	311111	1111	1	13	14	15			(12), (34)
8	5	3111	111311	3111	1	9	14	15			(23)
9	5	3111	111111	1113	3	11	13	14			(2, 3, 4)
10	5	1111	331311	1111	1	1	14	15			(1, 2, 3)
11	5	1111	311113	1111	3	3	12	15			(12), (34), (13)(24)
12	5	1111	311111	1133	1	3	13	14			(12), (34)
13	4	4400	400000	0000	0	12	13	14	15		(12), (34), [3], [4]
14	4	4000	000400	4000	0	8	9	14	15		(23), [23], [4]
15	4	4000	000000	0004	4	9	10	12	15		(2, 3, 4), [23], [24]
16	4	0000	440400	0000	0	0	1	14	15		(1, 2, 3), [123], [4]
17	4	0000	400004	0000	4	0	3	12	15		(12), (34), (13)(24), [12], [34]
18	4	0000	400000	0044	0	0	3	13	14		(12), (34), [123], [34]

T Remarks

1 0, S<sub>0</sub>16 FS, M, S<sub>1</sub>

32 3, FS, M

48 FS

32 FS

8 S<sub>0</sub>, 4

96 FS, M

192 FS

64 FS

64

48

96

24 2, M

48 3, FS

16 FS

16 3

FS

FS

N	0	1234					Standard Sum	Symmetry	T	Remarks
			111223	22333	34444	4				
19	4	4222	222000	0002	2	11 13 14 15	(2, 3, 4)	64	FS, M	
20	4	4220	220022	0220	0	10 13 14 15	(23)[4]	192	FS	
21	4	4200	200220	2202	2	8 13 14 15	(34)	192	FS	
22	4	2222	400000	2222	0	3 13 14 15	(12), (34)	96		
23	4	2222	000000	2222	4	7 11 13 14	(1, 2, 3, 4)	16	S <sub>3</sub>	
24	4	2220	402022	2000	2	2 13 14 15	(12)	192		
25	4	2220	002022	2400	2	6 10 13 15	(12)	192		
26	4	2200	422220	0022	0	0 13 14 15	(12), (34)	96	FS	
27	4	2200	022224	0022	0	4 11 12 15	(12)[34], (34)	96		
28	4	2200	022220	4022	0	4 9 14 15	(12)[4]	192		
29	4	2200	022220	0022	4	7 8 13 14	(12)[34], (34)	96		
30	4	2000	222400	2220	2	0 9 14 15	(23)	192		
31	4	2000	222000	2224	2	0 11 13 14	(2, 3, 4)	64		
32	3	5311	311111	1111	1	11 12 13 14 15	(34)	192	FS, M	
33	3	5111	111311	3111	1	10 11 12 13 15	(23)	192	FS	
34	3	5111	111111	1113	3	8 11 13 14 15	(2, 3, 4)	64	FS	
35	3	3311	511111	1111	1	3 12 13 14 15	(12), (34)	96	FS	
36	3	3111	111511	3111	1	7 8 9 14 15	(23)	192		
37	3	3111	111311	5111	1	7 10 11 12 13	(23)	192		
38	3	3111	111111	1115	3	7 9 10 12 15	(2, 3, 4)	64		
39	3	3111	111111	1113	5	7 8 11 13 14	(2, 3, 4)	64		
40	3	1111	531311	1111	1	2 3 12 13 15	(12)	192		
41	3	1111	511113	1111	3	1 2 13 14 15	(12), (34)	96		
42	3	1111	511111	1133	1	0 3 13 14 15	(12), (34)	96		
43	3	1111	113111	5113	1	3 5 8 14 15	(23)	192		
44	3	1111	311113	1111	5	5 6 9 10 15	(12), (34), (13)(24)	48		
45	3	3333	111111	1111	3	7 11 13 14 15	(1, 2, 3, 4)	16	S <sub>3, 4</sub> , M	
46	3	3331	113111	1113	1	6 11 13 14 15	(23)	192		
47	3	3311	131133	1111	1	5 10 13 14 15	(12)(34)	192		
48	3	3311	133111	1111	3	4 11 13 14 15	(34)	192		
49	3	3311	131111	1331	1	5 11 12 14 15	—	384		
50	3	3311	111111	3311	3	7 11 12 13 14	(12), (34)	96	FS	
51	3	3111	333111	1113	1	0 11 13 14 15	(2, 3, 4)	64		
52	3	3111	331131	1131	1	1 11 12 14 15	—	384		
53	3	3111	311331	1113	1	3 11 12 13 14	(34)	192	FS	
54	3	3111	311311	1311	3	3 10 12 13 15	—	384		
55	3	3111	311111	3313	1	3 8 13 14 15	(34)	192		
56	3	1111	331131	3311	1	6 7 9 11 12	(12)(34)	192		
57	3	1111	331111	3113	3	6 7 8 11 13	(23)	192		
58	3	1111	111111	3333	3	0 7 11 13 14	(1, 2, 3, 4)	16	S <sub>0, 3</sub>	
59	2	6220	220200	2000	0	10 11 12 13 14 15	(23), [4]	96	3, FS, M	
60	2	6200	200002	0022	2	8 11 12 13 14 15	(34), [34]	96	FS	
61	2	6000	000222	2220	0	9 10 11 12 13 14	(2, 3, 4), [234]	32	FS	
62	2	2220	620200	2000	0	2 3 12 13 14 15	(12), [4]	96	3	
63	2	2220	220200	6000	0	6 7 10 11 12 13	(1, 2, 3), [4]	32	3	
64	2	2200	600002	0022	2	1 2 12 13 14 15	(12), (34), [34]	48	FS	
65	2	2200	200006	0022	2	4 7 8 11 12 15	(12), (34), [34]	48	FS	
66	2	2200	200002	0062	2	4 7 9 10 13 14	(34), [34]	96	FS	



N	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Standard Sum	Symmetry	T	Remarks
67	2	2200	200002	0022	6	5	6	9	10	12	15	(12), (34), [34]	48	FS						
68	2	2000	000622	2220	0	1	6	8	9	14	15	(23), [234]	96							
69	2	2000	000222	6220	0	1	6	10	11	12	13	(23), [234]	96							
70	2	0000	622222	0000	2	1	2	3	12	13	14	(12), (34), [1234]	48							
71	2	0000	222222	0000	6	3	5	6	9	10	12	(1, 2, 3, 4), [1234]	8	S <sub>2</sub>						
72	2	4422	200002	2200	2	7	11	12	13	14	15	(12), (34)	96	FS, M						
73	2	4420	202020	2022	0	6	11	12	13	14	15	(12)[4]	192							
74	2	4400	222222	0000	2	4	11	12	13	14	15	(12)[34], (34)	96							
75	2	4222	400222	0002	0	3	11	12	13	14	15	(34)	192	FS						
76	2	4222	000222	4002	0	7	10	11	12	13	15	(23)	192							
77	2	4222	000222	0002	4	7	8	11	13	14	15	(2, 3, 4)	64							
78	2	4220	402200	0220	2	2	11	12	13	14	15	—	384							
79	2	4220	002240	0220	2	7	8	10	13	14	15	—	384							
80	2	4220	002200	4220	2	6	10	11	12	13	15	(23)	192							
81	2	4220	002200	0224	2	6	8	11	13	14	15	(23)	192							
82	2	4200	422002	2202	0	0	11	12	13	14	15	(34)	192							
83	2	4200	022402	2202	0	4	10	11	12	13	15	—	384							
84	2	4200	022002	2242	0	4	9	10	13	14	15	(34)	192							
85	2	4200	022002	2202	4	4	8	11	13	14	15	(34)	192							
86	2	4000	222222	4000	2	0	10	11	12	13	15	(23)	192							
87	2	4000	222222	0004	2	0	8	11	13	14	15	(2, 3, 4)	64							
88	2	2220	224240	2000	0	5	7	8	10	14	15	(12)[4]	192							
89	2	2220	224200	2400	0	5	7	10	11	12	14	—	384							
90	2	2220	224200	2004	0	0	6	11	13	14	15	(23)	192							
91	2	2220	224200	2000	4	2	4	11	13	14	15	(23)	192							
92	2	2220	220200	2440	0	3	4	10	13	14	15	(23)[4]	192							
93	2	2220	220200	2400	4	2	7	11	12	13	14	(12)	192							
94	2	2200	244002	0022	2	0	4	11	13	14	15	(34)	192							
95	2	2200	240402	0022	2	6	7	8	9	12	15	(12)[34]	192							
96	2	2200	240042	0022	2	0	5	10	13	14	15	(12)(34)	192							
97	2	2200	240002	4022	2	6	7	8	11	12	13	—	384							
98	2	2200	240002	0422	2	1	4	10	13	14	15	—	384							
99	2	2200	200002	4422	2	3	4	8	13	14	15	(12), (34)	96							
100	2	2000	440222	2220	0	0	1	10	13	14	15	(23)[4]	192							
101	2	2000	400222	2224	0	0	3	8	13	14	15	(34)	192							
102	2	2000	400222	2220	4	1	2	8	13	14	15	(34)	192							
103	2	2000	000222	2224	4	0	7	9	10	12	15	(2, 3, 4)	64							
104	2	0000	222222	4400	2	2	3	4	8	13	15	(12), (34)[12]	96							
105	2	2222	222222	2222	2	0	7	11	13	14	15	(1, 2, 3, 4)	16	S <sub>0</sub> , 3, 4						
106	2	2222	222222	2222	2	1	6	11	13	14	15	(23)	192							
107	2	2222	222222	2222	2	3	5	10	13	14	15	(12)(34)	192							
108	2	2222	222222	2222	2	3	7	11	12	13	14	(12), (34), (13)(24)	48	FS						
109	1	7111	111111	1111	1	9	10	11	12	13	14	15	(2, 3, 4)	64	FS, M					
110	1	1111	711111	1111	1	1	2	3	12	13	14	15	(12), (34)	96	FS					
111	1	1111	111111	7111	1	1	6	7	10	11	12	13	(1, 2, 3)	64						
112	1	1111	111111	1111	7	3	5	6	9	10	12	15	(1, 2, 3, 4)	16	S <sub>2</sub> , 4					
113	1	5331	111111	3111	1	7	10	11	12	13	14	15	(23)	192	M					
114	1	5311	131311	1111	1	5	10	11	12	13	14	15	—	384						

N	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Standard Sum	Symmetry	T	Remarks
115	1	5	3	1	1	1	1	1	1	3	7	8	11	12	13	14	15	(34)	192	
116	1	5	3	1	1	1	1	1	1	3	7	9	10	12	13	14	15	(34)	192	
117	1	5	1	1	1	1	1	1	1	3	1	1	10	11	12	13	14	(23)	192	FS
118	1	5	1	1	1	1	1	1	1	3	8	11	12	13	14	15	(34)	192		
119	1	5	1	1	1	1	1	1	1	3	7	8	9	10	13	14	15	(34)	192	
120	1	5	1	1	1	1	1	1	1	3	9	10	12	13	14	15	(34)	192		
121	1	5	1	1	1	1	1	1	1	3	7	9	10	11	12	13	14	(2, 3, 4)	64	
122	1	3	3	3	1	1	1	1	1	5	6	7	10	11	12	13	15	(1, 2, 3)	64	
123	1	3	3	1	1	5	1	3	1	1	4	5	10	11	13	14	15	—	384	
124	1	3	3	1	1	1	1	1	5	1	5	6	9	10	13	14	15	(12), (34)	96	
125	1	3	3	1	1	1	1	1	1	5	4	7	9	10	13	14	15	(34)	192	
126	1	3	3	1	1	1	1	1	1	5	4	7	8	11	13	14	15	(12), (34)	96	
127	1	3	1	1	1	5	3	1	1	1	2	3	9	12	13	14	15	—	384	
128	1	3	1	1	1	5	1	1	1	3	0	3	11	12	13	14	15	(34)	192	
129	1	3	1	1	1	1	3	5	1	1	3	5	10	11	12	13	14	(23)	192	
130	1	3	1	1	1	1	1	5	3	1	2	5	10	11	12	13	15	—	384	
131	1	3	1	1	1	5	1	1	1	1	3	1	2	11	12	13	14	(34)	192	
132	1	3	1	1	1	3	3	1	1	1	6	7	9	10	11	12	13	(23)	192	
133	1	3	1	1	1	3	1	1	1	3	4	7	9	10	11	12	15	(34)	192	
134	1	3	1	1	1	1	1	1	3	3	0	7	10	11	12	13	15	(23)	192	
135	1	3	1	1	1	1	3	1	1	1	3	1	7	10	11	12	13	(23)	192	
136	1	3	1	1	1	1	1	1	1	5	3	6	10	11	12	13	15	(23)	192	
137	1	3	1	1	1	1	1	1	1	1	5	5	6	9	10	11	12	(34)	192	
138	1	1	1	1	1	5	3	1	1	3	4	5	7	8	10	11	15	(12)(34)	192	
139	1	1	1	1	1	5	3	3	1	1	3	4	5	6	9	10	11	(34)	192	
140	1	1	1	1	1	5	3	1	1	1	4	5	7	9	10	11	14	—	384	
141	1	1	1	1	1	5	1	1	1	1	3	5	6	7	9	10	11	(12), (34)	96	FS
142	1	1	1	1	1	3	1	3	1	1	2	3	4	9	13	14	15	—	384	
143	1	1	1	1	1	3	1	1	1	1	3	2	3	5	9	12	14	(12)	192	
144	1	1	1	1	1	3	3	3	1	1	5	1	2	4	11	13	14	(2, 3, 4)	64	
145	1	1	1	1	1	3	1	1	1	1	5	1	2	7	11	12	13	(12), (34)	96	
146	1	3	3	3	3	3	1	1	1	1	3	7	11	12	13	14	15	(12), (34), (13)(24)	48	FS, M
147	1	3	3	3	1	3	1	1	1	1	3	7	10	12	13	14	15	—	384	
148	1	3	3	3	1	1	1	1	1	1	3	2	7	11	12	13	14	(12)	192	
149	1	3	3	1	1	3	3	1	1	3	3	7	8	12	13	14	15	(34)	192	
150	1	3	3	1	1	3	3	1	1	3	3	6	9	12	13	14	15	(12)(34)	192	
151	1	3	3	1	1	3	3	1	1	1	2	7	9	12	13	14	15	—	384	
152	1	3	3	1	1	3	3	1	1	3	0	7	11	12	13	14	15	(12), (34)	96	
153	1	3	3	1	1	3	3	1	1	3	6	7	9	11	12	13	14	(12)(34)	192	
154	1	3	3	1	1	3	3	1	1	3	6	7	8	11	12	13	15	—	384	
155	1	3	3	1	1	3	3	1	1	3	6	7	9	10	12	13	15	—	384	
156	1	3	1	1	1	3	3	1	3	1	6	7	8	9	11	12	15	—	384	
157	1	3	1	1	1	3	3	1	3	1	6	7	8	9	11	13	14	(23)	192	
158	1	3	1	1	1	3	3	1	3	1	4	7	8	10	11	13	15	—	384	
159	1	3	1	1	1	3	3	3	3	1	0	7	8	11	13	14	15	(2, 3, 4)	64	
160	1	3	1	1	1	3	3	1	1	3	5	6	8	9	11	14	15	—	384	
161	1	3	1	1	1	3	3	3	1	3	1	6	8	11	13	14	15	(23)	192	
162	1	1	1	1	1	3	3	3	3	1	0	1	6	11	13	14	15	(23)	192	
163	1	1	1	1	1	3	3	1	3	1	0	3	5	10	13	14	15	(12)(34)	192	

N	0	1234	111223				111223				Standard Sum	Symmetry	T	Remarks		
			234344	3444	4	1	2	3								
164	1	1111	311113	33333	1	0	3	7	11	12	13	14	(12), (34), (13)(24)	48		
165	0	8000	000000	0000	0	8	9	10	11	12	13	14	15	(2, 3, 4), [2], [3], [4]	8	1, SD, M
166	0	0000	800000	0000	0	4	5	6	7	8	9	10	11	(12), (34), [12], [3], [4]	12	2
167	0	0000	000000	8000	0	2	3	4	5	8	9	14	15	(1, 2, 3), [12], [13], [4]	8	3, FS, SD
168	0	0000	000000	0000	8	1	2	4	7	8	11	13	14	(1, 2, 3, 4), [12], [13], [14]	2	FS, S <sub>1</sub> , s
169	0	6222	000000	2222	0	7	9	10	11	12	13	14	15	(2, 3, 4)	64	SD, M
170	0	6220	002022	2000	2	6	9	10	11	12	13	14	15	(23)	192	
171	0	6200	022220	0022	0	4	9	10	11	12	13	14	15	(34)	192	FS
172	0	6000	222000	2220	2	0	9	10	11	12	13	14	15	(2, 3, 4)	64	FS
173	0	2222	000000	6222	0	1	6	7	10	11	12	13	15	(1, 2, 3)	64	SD
174	0	2220	006022	2000	2	2	4	6	9	11	13	14	15	(23)	192	
175	0	2220	002022	6000	2	1	6	7	10	11	12	13	14	(1, 2, 3)	64	
176	0	2220	002022	2000	6	3	5	6	9	10	12	14	15	(1, 2, 3)	64	
177	0	2200	062220	0022	0	1	4	5	10	11	12	14	15	—	384	
178	0	2200	022220	0062	0	3	4	7	9	10	12	13	14	(34)	192	
179	0	2000	622000	2220	2	1	2	3	8	12	13	14	15	(34)	192	FS
180	0	2000	222000	6220	2	1	6	7	8	10	11	12	13	(23)	192	FS
181	0	2000	222000	2220	6	3	5	6	8	9	10	12	15	(2, 3, 4)	64	FS
182	0	4440	000000	4000	0	6	7	10	11	12	13	14	15	(1, 2, 3), [4]	32	3, SD, M
183	0	4400	040400	0000	0	6	7	8	9	12	13	14	15	(12)[3], [4]	96	3
184	0	4400	000004	0000	4	5	6	9	10	12	13	14	15	(12), (34), [34]	48	FS
185	0	4400	000000	0044	0	5	6	8	11	12	13	14	15	(12)[3], (34), [34]	48	FS, SD
186	0	4000	440000	4000	0	0	1	10	11	12	13	14	15	(23), [4]	96	3, FS
187	0	4000	400004	0004	0	0	3	8	11	12	13	14	15	(34), [34]	96	FS
188	0	4000	000440	0040	0	0	7	8	9	10	13	14	15	(34), [234]	96	
189	0	4000	400000	0040	4	0	3	9	10	12	13	14	15	(34), [34]	96	FS
190	0	4000	000000	4440	0	0	7	9	10	11	12	13	14	(2, 3, 4), [234]	32	FS, SD
191	0	0000	440044	0000	0	0	1	2	5	10	13	14	15	(14)[3], (23)[4], (12)(34), [1234]	24	
192	0	0000	444000	0000	4	0	1	2	4	11	13	14	15	(2, 3, 4), [1234]	32	FS
193	0	0000	444000	0440	0	0	1	2	5	11	12	14	15	(23)[4], [123]	96	FS
194	0	0000	400000	4400	4	0	1	2	7	11	12	13	14	(12), (34), [12]	48	FS
195	0	4422	022220	2200	0	5	7	10	11	12	13	14	15	(12)(34)	192	M
196	0	4420	020202	2022	2	4	7	10	11	12	13	14	15	—	384	
197 a	0	4222	222400	0002	2	3	5	10	11	12	13	14	15	(23)	192	
197 b	0	4222	222400	0002	2	6	7	8	9	11	13	14	15	(23)	192	
198 a	0	4222	222000	4002	2	1	7	10	11	12	13	14	15	(23)	192	
198 b	0	4222	222000	4002	2	6	7	9	10	11	12	13	15	(23)	192	
199	0	4220	224022	0220	0	2	6	9	11	12	13	14	15	—	384	
200 a	0	4220	220422	0220	0	2	5	10	11	12	13	14	15	(23)[4]	192	
200 b	0	4220	220422	0220	0	6	7	8	9	11	12	14	15	(23)[4]	192	
201 a	0	4220	220022	4220	0	0	7	10	11	12	13	14	15	(23)	192	
201 b	0	4220	220022	4220	0	6	7	9	10	11	12	13	14	(23)	192	
202	0	4220	220022	0224	0	3	6	8	11	12	13	14	15	—	384	
203	0	4220	220022	0220	4	2	7	8	11	12	13	14	15	—	384	

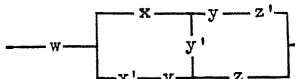
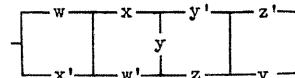
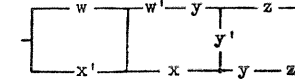
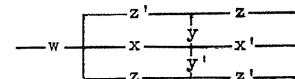
N	0	1234															Standard Sum	Symmetry	T	Remarks				
			111223				1112																	
			234344				2233																	
			3444				3444																	
204 a	0	4200	$\bar{2}\bar{4}02\bar{2}0$	$\bar{2}\bar{2}0\bar{2}$	$\bar{2}$	0	5	10	11	12	13	14	15	—		384								
204 b	0	4200	$240\bar{2}\bar{2}0$	$\bar{2}\bar{2}0\bar{2}$	$\bar{2}$	6	7	8	9	11	12	13	14	—		384								
205 a	0	4200	$\bar{2}00\bar{2}\bar{2}\bar{4}$	$\bar{2}\bar{2}0\bar{2}$	$\bar{2}$	0	7	8	11	12	13	14	15	(34)		192								
205 b	0	4200	$200224$	$\bar{2}\bar{2}0\bar{2}$	$\bar{2}$	5	6	9	10	11	12	13	14	(34)		192								
206 a	0	4200	$\bar{2}00220$	$22\bar{4}\bar{2}$	$\bar{2}$	3	4	9	10	12	13	14	15	(34)		192								
206 b	0	4200	$200\bar{2}\bar{2}0$	$22\bar{4}\bar{2}$	$\bar{2}$	4	7	8	9	10	13	14	15	(34)		192								
207	0	2222	$\bar{4}40000$	$2\bar{2}\bar{2}\bar{2}$	0	2	3	7	9	12	13	14	15	—		384								
208 a	0	2222	$\bar{4}0000\bar{4}$	$\bar{2}\bar{2}\bar{2}\bar{2}$	0	0	3	7	11	12	13	14	15	(12), (34), (13)(24)		48								
208 b	0	2222	$400004$	$\bar{2}\bar{2}\bar{2}\bar{2}$	0	5	6	7	9	10	11	13	14	(12), (34), (13)(24)		48								
209 a	0	2222	$\bar{4}00000$	$\bar{2}\bar{2}\bar{2}\bar{2}$	4	1	2	7	11	12	13	14	15	(12), (34)		96								
209 b	0	2222	$400000$	$\bar{2}\bar{2}\bar{2}\bar{2}$	$\bar{4}$	5	6	7	9	10	11	12	15	(12), (34)		96								
210	0	2220	$\bar{4}420\bar{2}\bar{2}$	$2000$	2	2	3	7	8	12	13	14	15	—		384								
211 a	0	2220	$\bar{4}0\bar{2}0\bar{2}\bar{2}$	$\bar{2}\bar{4}00$	2	0	2	7	11	12	13	14	15	(12)		192								
211 b	0	2220	$402022$	$\bar{2}\bar{4}00$	$\bar{2}$	5	6	7	9	10	11	12	14	(12)		192								
212 a	0	2220	$\bar{4}020\bar{2}\bar{2}$	$\bar{2}0\bar{4}0$	$\bar{2}$	0	3	7	10	12	13	14	15	—		384								
212 b	0	2220	$40\bar{2}022$	$\bar{2}0\bar{4}0$	2	4	6	7	9	10	11	13	14	—		384								
213	0	2220	$0020\bar{2}\bar{2}$	$2\bar{4}40$	2	2	5	7	8	11	12	14	15	—		384								
214 a	0	2200	$\bar{4}\bar{2}\bar{2}22\bar{4}$	$00\bar{2}\bar{2}$	0	0	3	4	11	12	13	14	15	(34), (12)[34]		96								
214 b	0	2200	$422\bar{2}\bar{2}4$	$00\bar{2}\bar{2}$	0	5	6	7	8	9	10	13	14	(34), (12)[34]		96								
215 a	0	2200	$\bar{4}\bar{2}\bar{2}\bar{2}20$	$\bar{4}02\bar{2}$	0	0	1	6	11	12	13	14	15	(12)[4]		192								
215 b	0	2200	$4222\bar{2}0$	$\bar{4}02\bar{2}$	0	5	6	7	8	10	11	12	13	(12)[4]		192								
216	0	2200	$02\bar{2}\bar{2}2\bar{4}$	$\bar{4}02\bar{2}$	0	0	6	7	8	11	12	13	15	—		384								
217 a	0	2200	$\bar{4}\bar{2}\bar{2}220$	$0022$	4	1	2	4	11	12	13	14	15	(34), (12)[34]		96								
217 b	0	2200	$422\bar{2}\bar{2}0$	$0022$	$\bar{4}$	5	6	7	8	9	10	12	15	(34), (12)[34]		96								
218 a	0	2200	$0\bar{2}\bar{2}\bar{2}\bar{4}$	$00\bar{2}\bar{2}$	4	0	4	7	8	11	13	14	15	(12), (34)		96								
218 b	0	2200	$022224$	$00\bar{2}\bar{2}$	$\bar{4}$	3	5	6	9	10	12	13	14	(12), (34)		96								
219	0	2200	$02\bar{2}\bar{2}20$	$\bar{4}\bar{4}\bar{2}\bar{2}$	0	0	6	7	9	11	12	13	14	(12)(34)		192								
220	0	2200	$0\bar{2}\bar{2}\bar{2}\bar{2}0$	$402\bar{2}$	4	2	4	5	8	11	13	14	15	—		384								
221	0	2000	$22\bar{2}\bar{4}\bar{4}0$	$\bar{2}\bar{2}\bar{2}0$	$\bar{2}$	0	6	7	8	9	10	13	15	—		384								
222 a	0	2000	$\bar{2}\bar{2}\bar{2}\bar{4}00$	$\bar{2}\bar{2}\bar{2}\bar{4}$	2	0	1	6	8	11	13	14	15	(23)		192								
222 b	0	2000	$222400$	$\bar{2}\bar{2}\bar{2}\bar{4}$	$\bar{2}$	3	5	6	8	10	11	12	13	(23)		192								

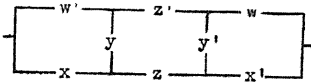
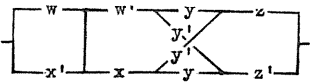
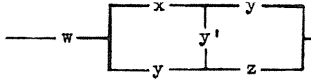
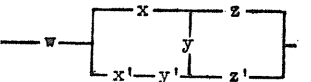
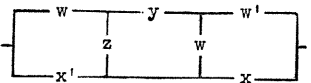
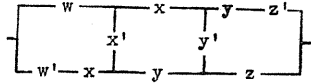
## Appendix 2. Table of Minimal Switching Circuits for Boolean Functions of four Variables

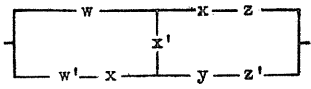
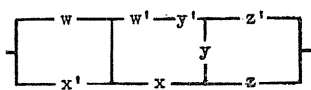
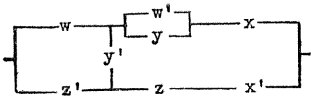
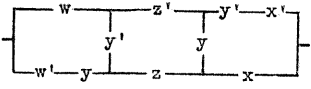
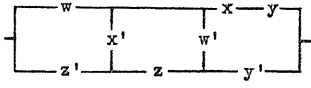
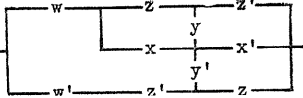
Table of Symbols for Vacuum Tube and Transistor Switching Circuits

Symbol	Switching Action	Vacuum tube gate	NPN Transistor gate	PNP Transistor gate
$P_1(e)$	$e'$	Complement Gate	Complement Gate	—
$P_n(e_1, \dots, e_n)$	$e'_1 \dots e'_n$	Triode Gate	Series Gate	Parallel Gate
$S_1(e)$	$e'$	—	—	Complement Gate
$S_n(e_1, \dots, e_n)$	$e'_1 + \dots + e'_n$	Pentode Gate ( $n=2$ )	Parallel Gate	Series Gate
$F_n(e_1, \dots, e_n)$	$e_1 + \dots + e_n$	Cathode Follower Gate	Emitter Follower Gate	—
$B_n(e_1, \dots, e_n)$	$e_1 \dots e_n$	—	—	Emitter Follower Gate
.	Multipli- cation	Common Plate Resistor Connec- tion	Common Collector Resistor Connec- tion	—
+	Addition	--	—	Common Collector Resistor Connec- tion

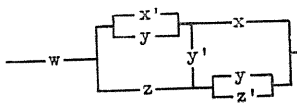
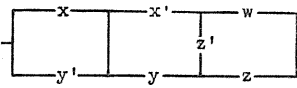
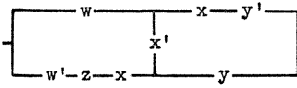
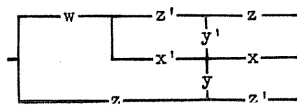
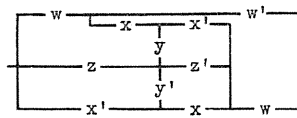
N	H	Switching Circuit	R	C	S	G	T
1	0	0	0	0	0	0	0
2	1	$wxyz$ $P_1(w', x', y', z')$	4	4	8	4	4
3	2	$wxy$ $P_3(w', x', y')$	3	3	6	3	3
4	3	$wxyz + wx'y'z'$ $wx(yz + y'z')$ $P_2(w', x')S_2(y, z')S_2(y', z)$	10*	6	10	6	6
5	4	$wxyz + wx'y'z'$ $w(xy + x'z')(y' + z)$ $P_1(w')S_2(x, y')S_2(x', z)S_2(y, z')$	10	7	11	7	7
6	5	$wxyz + w'x'y'z'$ $(wx + w'y')(yz + x'z')$ $S_2(w, x')S_2(w', z)S_2(x, y')S_2(y, z')$	10	8	12	8	8
7	6	$wxy + wxz$ $wx(y + z)$ $P_2(w', x')S_2(y', z')$	8*	4	8	4	4

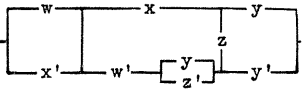
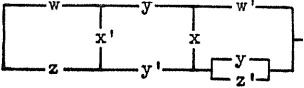
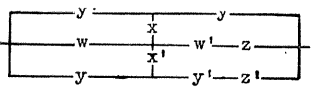
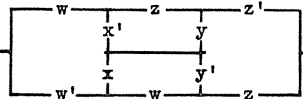
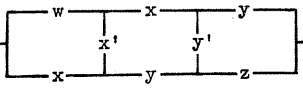
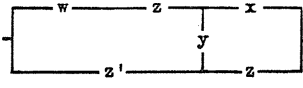
N	H	Switching Circuit	R	C	S	G	T
8	7	$wxy + wx'y'z$ $w(xy + x'zy')$ $P_1(w')S_2(x, y')S_2(x', y)S_2(x', z')$	9		6 10		7 7
9	9	$wxyz' + wxy'z + wx'yz$  $P_1(w')S_2(x, p)S_2[x', P_1(p)]S_2(y', z'), p = P_2(y', z')$	15*		8 13		10 10
10	8	$wxy + w'x'y'z$ $(wx + w'zy')(x' + y)$ $S_2(w', x)S_2(w, y')S_2(w', z')S_2(x', y)$	9		7 11		8 8
11	10	$wxyz + wxy'z' + w'x'yz$  $S_2(w, x')S_2(w', x)S_2(w', y')S_2(y, z')S_2(y', z)$	15		9 14		10 10
12	11	$wxyz' + wxy'z + w'x'yz$  $S_2(w, x')S_2(y', z')S_2(w', p)S_2[x, P_1(p)], p = S_2(y, z)$	15		9 14		11* 11
13	12	$wx$ $P_2(w', x')$	2	2	4		2 2
14	19	$wxy + wx'y'$ $w(xy + x'y')$ $P_1(w')S_2(x, y')S_2(x', y)$	8*		5 8		5 5
15	25	$wxyz + wxy'z' + wx'yz' + wx'y'z$  $P_1(w')S_2(x, p)S_2[x', P_1(p)], p = S_2(y, z)S_2(y', z')$	20*		9 14		10 10
16	24	$wxy + w'x'y'$ $(wx + w'y')(x' + y)$ $S_2(w, x')S_2(w', y)S_2(x, y')$	8		6 9		6 6
17	29	$wxyz + wxy'z' + w'x'yz + w'x'y'z'$ $(wx + w'x')(yz + y'z')$ $S_2(w, x')S_2(w', x)S_2(y, z')S_2(y', z)$	20*		8 12		8 8

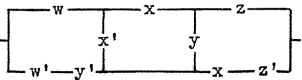
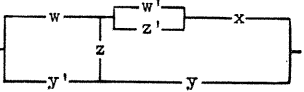
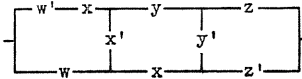
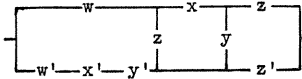
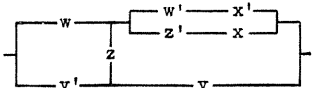
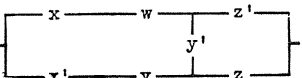
N	H	Switching Circuit	R	C	S	G	T
18	30	$wxyz' + wxy'z + w'x'yz + w'x'y'z'$	20*				
				8	16		
				10	15		
		$S_2(w, x') S_2(w', p) S_2[x, P_1(p)], p = S_2(y, z) S_2(y', z')$				11	11
19	13	$wxy + wxz + wyz$	12*				
				6	11		
		$P_1(w') S_2(x', y') S_2(x', z') S_2(y', z')$				7	7
20	14	$wxz + wyz'$ $w(xz + yz')$ $P_1(w') S_2(x', z) S_2(y', z')$	8*	5	9	5	5
21	15	$wxy + wxz + wx'y'z'$	13*				
				7	11		
		$P_1(w') S_2(x, p) S_2[x', P_1(p)], p = P_2(y, z)$				8	8
22	16	$wxy + wxz + w'x'yz$	13				
				7	12		
		$S_2(w, x') S_2(y', z') S_2[w', F_3(x, y', z')]$				9	9
23	26	$wxyz' + wxy'z + wx'yz + w'xyz$	20				
				10	16		
		$S_2(p, q) P_1[P_2(p, q)],$ $p = P_2(w', x') S_2(y', z'), q = P_2(y', z') S_2(w', x')$				13*	13
24	17	$wxy + wxz + w'x'yz'$ $(w' + x)(w + x'z')(y + z)$ $S_2(w, x') S_2(w', x) S_2(w', z) S_2(y', z')$	13*	7	11	8	8

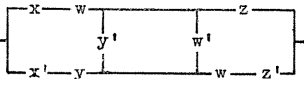
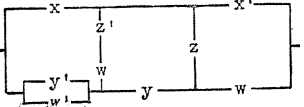
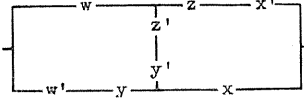
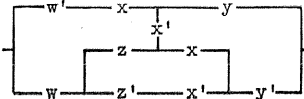
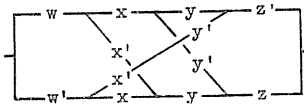
N	H	Switching Circuit	R	C	S	G	T
25	20	$wxz + wx'y'z' + w'xyz'$	14				
				8	13		
		$S_2(w', x') S_2(z, p) S_2[z', S_2(y, p)], p = S_2(w, x)$					10* 10
26	18	$wxy + wxz + w'x'y'z'$	13				
				8	12		
		$S_2(w', x) S_2(w, p) S_2[x', P_1(p)], p = P_2(y, z)$					9 9
27	22	$wyz + xy'z'$ $ywz + y'xz'$ $F_2[P_3(w', y', z'), P_3(x', y, z)]$ $P_1[S_3(w, y, z) S_3(x, y', z')]$	8	6	10	8	7
28	21	$wxy + wx'y'z + w'xy'z'$	14				
				8	12		
		$S_2(w', z) S_2(x', z') S_2(y, p) S_2[y', P_1(p)], p = S_2(w, x)$					11 11
29	27	$wxyz' + wxy'z + wx'y'z' + w'xyz$	20				
				10	16		
		$S_2(w', z') S_2(x', z) S_2(y', p) S_2[y, P_1(p)], p = P_1(x') S_2(w, z)$					12* 12
30	23	$wxy + wx'y'z + w'x'y'z'$	14				
				8	12		
		$F_2[P_3(w', x', y'), P_2(x, y) S_2(w, z') S_2(w', z)]$					11 11
31	28	$wxyz' + wxy'z + wx'y'z + w'x'y'z'$	20				
				10	15		
		$S_2(w', p) S_2[P_2(w', x), q] P_1[P_1(x') S_2(p, q)],$ $S_2(w', p) S_3(w, x', q) P_1[P_1(x') S_2(p, q)],$ $p = S_2(y', z'), q = S_2(y, z)$					14* 13

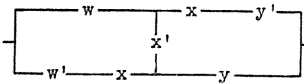
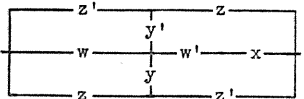
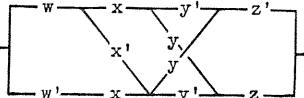
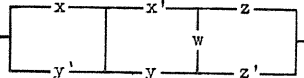
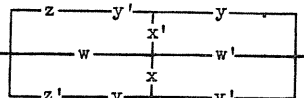


N	H	Switching Circuit	R	C	S	G	T
32	31	$wx + wyz$ $w(x + yz)$ $P_1(w') S_2(x', y') S_2(x', z')$	7		4 8	5 5	
33	38	$wxy' + wx'y + wxz$ $w(x + y)(x' + y' + z)$ $P_1(w') S_2(x', y') S_2[x, P_2(y', z)]$ $P_1(w') S_2(x', y') S_3(x, y, z')$	12*		6 10	7	6
34	33	$wxy + wxz + wyz + wx'y'z'$	17*				
				8	13		
		$S_2(x, p) P_2[w', P_2(x, p) S_2(y, z)], p = P_2(y, z)$				10 10	
35	32	$wx + w'x'yz$ $wx + w'yzx'$ $S_2(w, x') S_2[w', F_3(x, y', z')]$	8		6 10	7 7	
36	46	$wxy + wx'y' + xyz$	12				
				7	11		
		$S_2(w', x') S_2(w', z') S_2(x, y') S_2(x', y)$				8 8	
37	50	$wxy' + wx'y + w'xyz$	13				
				8	13		
		$S_2(x', y') S_2(w, p) S_2[w', S_2(z, p)], p = P_2(x', y')$				10 10	
38	51	$xyz + wxy'z' + wx'yz' + wx'y'z$	19				
				9	14		
		$S_2(w', x') S_2(z, p) S_2[z', S_2(w, p)], p = S_2(x, y) S_2(x', y')$ $S_2(y, q) P_1[P_1(y) S_2(w, q)], q = S_2(x, z) S_3(w, x', z')$				12	11
39	55	$wxyz' + wxy'z + wx'yz + wx'y'z' + w'xyz$	25				
				11	17		
		$S_2(w', y') S_2(w', z') S_2(x, p) S_2[x', P_1(p)],$ $p = P_1(w') S_2(y, z') S_2(y', z)$				14* 14	

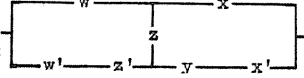
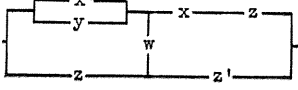
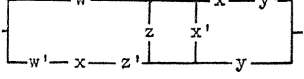
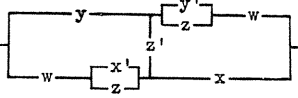
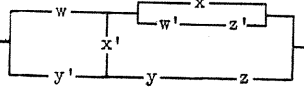
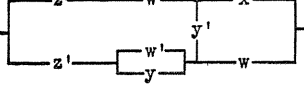
N	H	Switching Circuit	R	C	S	G	T
40	49	$wxy + wxz + w'x'y$ $(w + x')[w'y + x(y' + z)]$ $F_2[P_2(w', x') S_2(y, z'), P_3(w, x, y')]$	12		7 11		9 9
41	48	$wxy + wxz + w'x'yz' + w'x'y'z$ $(w' + x)[w + x'(y' + z')](y + z)$ $S_2(w, x') S_2(w', x) S_2(y', z') P_1[P_3(w, y', z')]$ $S_2(w, x') S_2(w', x) S_2(y', z') S_3(w', y, z)$	18*		8 12	10	9
42	47	$wxy + wxz + w'x'yz + w'x'y'z'$ 	18		9 14		
		$F_2[P_2(w', x') S_2(y', z'), P_2(w, x) S_2(y, z') S_2(y', z)]$				12 12	
43	52	$wxy + wx'y'z' + w'xy'z + w'x'yz$  	19		9 17		
		$S_2(w, p) P_1[S_2(w, y) S_2(w, z') S_2(z, p)], p = S_2(x, y) S_2(x', y')$			10 15*	13* 13	
44	57	$wxyz + wx'y'z' + wx'y'z + w'xyz' + w'xy'z$ 	25*		10 16		
		$S_2(w', x') S_2(y', z') S_2(p, q) P_1[P_2(p, q)],$ $p = S_2(w, x), q = P_2(y', z')$				13* 13	
45	34	$wxy + wxz + wyz + xyz$ 	16		8 14		
		$S_2(w', x') S_2(y', z') P_1[S_2(w, x) S_2(y, z)]$				9 9	
46	35	$wxz + wyz + xyz'$ 	12		6 11		
		$S_2(w', z) S_2(x', y') S_2(x', z') S_2(y', z')$				8 8	

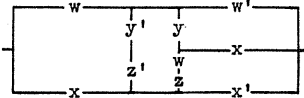
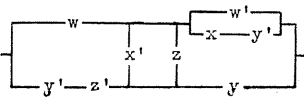
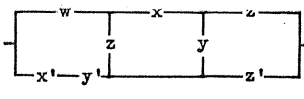
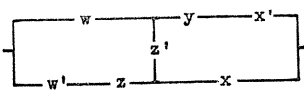
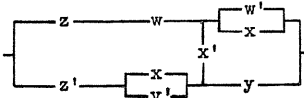
N	H	Switching Circuit	R	C	S	G	T
47	39	$wxy + wyz' + xy'z$ $(w + y')(y + z)(x + z')$ $S_2(w', y) S_2(x', z) S_2(y', z')$	12*		6 10		6 6
48	36	$wxy + wxz + wyz + w'xy'z'$  $S_2(x', y') S_2(x', z') S_2(w, p) S_2[w', P_1(p)]$ , $p = P_2(y, z)$	17*		9 14		11 11
49	40	$wxz' + wyz + w'xy'z$  $S_2(w', p) P_1[P_1(p) S_2(w', x)]$ , $p = S_2(y', z) S_2(x', z')$	13		7 11		10* 10
50	42	$wxy' + wxz' + wx'yz + w'xyz$  $S_2(p, q) P_1[P_2(p, q)]$ , $p = P_2(w', x')$ , $q = S_2(w', x') P_2(y', z')$	18		9 14		11 11
51	37	$wxy + wxz + wyz + w'x'y'z'$  $S_2(x, p) P_2[P_2(w, p), P_2(w', x) S_2(y, z)]$ , $p = P_2(y, z)$	17		9 15		12* 12
52	41	$wxz' + wyz + w'x'y'z$  $S_2(w', x) S_2(w', y) S_2(x', z') P_1[P_3(w', y, z')]$ $S_2(w', x) S_2(w', y) S_2(x', z') S_3(w, y', z)$	13		8 12*		10 9
53	45	$wxy' + wxz' + x'yz$  $S_2(x', p) S_2[x, S_2(w, p)]$ , $p = S_2(y, z)$	12		7 11		8* 8

N	H	Switching Circuit	R	C	S	G	T
54	44	$wxy' + wxz + wx'y'z + w'x'yz$ 	18*				
		$S_2(x, p) P_1[P_1(p) S_2(w, x)], p = P_1(y') S_2(w, z) S_2(w', z')$		9	14		11* 11
55	43	$wxy + wxz + wx'y'z' + w'x'yz$ 	18				
		$S_2(x, p) P_1[P_1(p) S_2(w, x)], p = S_2(w', y') S_2(w, z) S_2(y, z')$		9	14		12* 12
56	54	$w'xy + wx'z + wxy'z'$ 	13				
		$S_2(w', x') S_2(w', y') S_2(x', z') P_1[P_2(w', x') S_2(y', z')]$		8	12		11 11
57	53	$w'xy + wxy'z + wx'yz + wx'y'z'$ 	19				
		$S_2(w', p) S_2(z, q) P_1[P_1(w') S_2[p, F_2(z, q)]],$ $p = S_2(x, y), q = P_2(x, y)$		10	15		14* 14
58	56	$wxyz' + wxy'z + wx'yz + w'xyz + w'x'y'z'$ 	25				
		$S_2[p, P_1(q)] S_2[q, P_1(p)],$ $p = S_2(w', x') S_2(y, z), q = S_2(w, x') S_2(w', x) S_2(y', z')$		12	18		16* 16
59	58	$wx + wy$ $w(x + y)$ $P_1(w') S_2(x', y')$	6*				
				3	6		3 3
60	59	$wx + wyz + wy'z'$ $w(x + yz + y'z')$ $P_1(w') S_2[x', S_2(y, z) S_2(y', z')]$	11*				
				6	10		7 7
61	79	$wxy' + wx'z + wyz'$ $w(x + y + z)(x' + y' + z')$ $P_2[w', S_2(x, y') S_2(x', z) S_2(y, z')]$ $P_1(w') S_3(x, y, z) S_3(x', y', z')$	12*				
				7	11		8 7

N	H	Switching Circuit	R	C	S	G	T
62	66	$wx + w'x'y$ $wx + w'yx'$ $S_2(w, x') S_2(w', x) S_2(w', y')$	7		5 8		6 6
63	101	$wxy' + wx'y + w'xy$  $S_2(w', x') S_2(y', p) S_2[y, P_1(p)], p = S_2(w, x)$	12		7 11		9 9
64	67	$wx + w'x'yz' + w'x'y'z$ $(w' + x)[w + x'(y + z)(y' + z')]$ $F_2[P_2(w', x'), P_2(w, x) S_2(y, z) S_2(y', z')]$	13		8 12		10 10
65	98	$wyz + wy'z' + xyz + xy'z'$ $(w + x)(yz + y'z')$ $S_2(w', x') S_2(y, z') S_2(y', z)$	16*		6 10		6 6
66	102	$wyz' + wy'z + w'xyz + w'xy'z'$  $S_2(w, p) S_2[w', S_2(x, p)], p = S_2(y, z') S_2(y', z)$	18		9 14		10 10
67	106	$wxyz + wxy'z' + wx'yz' + wx'y'z + w'xyz' + w'xy'z$  $S_2(p, q) P_1[P_2(p, q)],$ $p = P_2(w', x'), q = S_2(w', x') S_2(y, z) S_2(y', z')$	30*		11 17		13 13
68	99	$wxy + wx'y' + xyz' + x'y'z$  $F_2[P_2(x', y') S_2(w', z), P_2(x, y) S_2(w', z')]$	16*		7 11		10 10
69	103	$wxy' + wx'y + w'xyz' + w'x'y'z$  $F_2[P_1(w') S_2(x, y) S_2(x', y'), P_1(w) S_2(x', y) S_2(x, z) S_2(y', z')]$	18		10 14		14 14

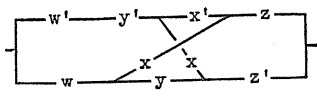
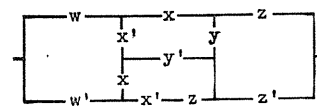
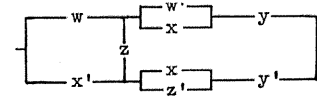
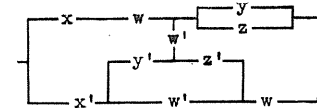
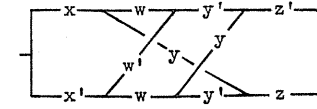
N	H	Switching Circuit	R	C	S	G	T
70	100	$wxy' + w'x'y + wxz' + w'x'z$ $(w+x')[w'(y+z) + x(y'+z')]$ $F_2[P_2(w, x)S_2(y', z'), P_2(w', x')S_2(y, z)]$	16*				
				8	12		10 10
71	107	$wxy'z' + wx'y'z + wx'yz' + w'xyz' + w'xy'z + w'x'yz$	30				
				12	18		
		$S_2[p, P_1(q)]S_2[q, P_1(p)],$ $p = S_2(w, x)S_2(y, z), q = S_2(w', x')S_2(y', z')$				14*	14
72	60	$wx + wyz + xyz$	11				
				6	11		
		$S_2(w', x')P_1[S_2(w, x)S_2(y, y)]$				7*	7
73	61	$wx + wyz + xyz'$	11				
				5	9		
		$P_2[P_1(w)S_2(y, z'), P_1(x)S_2(y, z)]$				8	8
74	62	$wx + wyz + xy'z'$ $(w+y'z')(x+zy)$ $P_2[P_1(w)S_2(y', z'), P_1(x)S_2(y, z)]$	11				
				6	11		8 8
75	63	$wx + x'yz$ $wx + yzx'$ $S_2(w', x)S_2(x', y')S_2(x', z')$	7				
				5	9		6 6
76	73	$wxy' + wx'y + xyz$	12				
				7	12		
		$S_2(x', y')S_2(w', p)S_2[z', P_1(p)], p = S_2(x, y)$				9*	9
77	68	$wxy + wxz + wyz + xyz + wx'y'z'$	21				
				9	15		
		$S_2(w', x')S_2(x, p)P_1[P_1(p)S_2(w, x)S_2(y, z)], p = P_2(y, z)$				12*	12

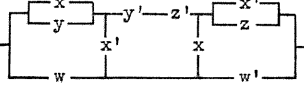
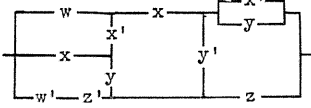
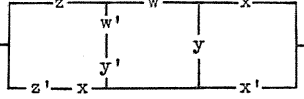
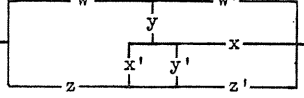
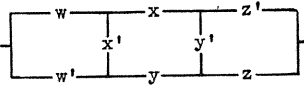
N	H	Switching Circuit	R	C	S	G	T
78	64	$wx + wyz + w'x'y'z'$	12				
				7	11		
		$S_2(x', y') S_2(w, p) S_2[w', P_1(p)], p = P_2(x, z)$				9	9
79	74	$wxz + wx'z' + wyz' + xyz$	16*				
				7	12		
		$F_2[P_2(w', z) S_2(x, y'), P_2(x', z') S_2((w', y'))]$				10	
		$S_2(w', y') S_2(w', z') S_2(x', z) S_3(x, y', z')$					9
80	81	$wxy' + wx'y + wxz + w'xyz'$	17				
				9	14		
		$S_2(w, p) S_2[w', P_1(p)] S_2(x', y'), p = P_3(x', y'z)$				10	10
81	69	$wxz + wyz + xyz' + wx'y'z'$	17				
				9	15		
		$P_2[S_2(w, y') S_2(w, z) S_2(x, z'), P_1(y) S_2(x, z) S_2(x', z')]$				13	13
82	65	$wx + wyz + w'x'y'z'$	12				
				8	12		
		$P_2[P_1(w) S_2(x', z'), P_1(x) S_2(w', y') S_2(y, z)]$				10*	10
83	82	$wx'y + wxz + xy'z'$ $(x+y)[w(x'+z) + y'z']$ $S_2(x', y') S_2(w', p) S_2[y, P_1(p)], p = S_2(x, z')$	12				
				7	12		
						9	9
84	80	$wxy + wyz' + wy'z + w'xy'z'$	17				
				8	13		
		$S_2(w', p) P_1[P_1(p) S_2(w', x)], p = S_2(y', z') P_1[P_3(x, y', z')]$				12*	
		$S_2(w, q) S_2[w', S_2(x, q)] S_3(x', y, z), q = P_2(y, z)$					11

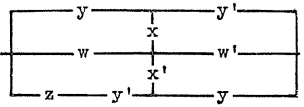
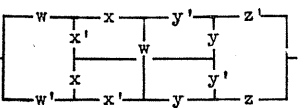
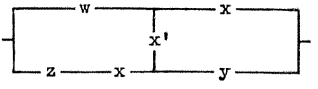
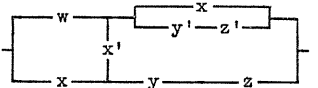
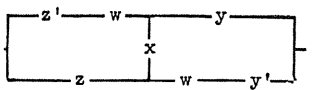
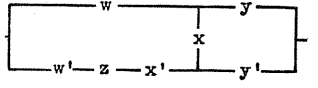
N	H	Switching Circuit	R	C	S	G	T
85	70	$wxy + wxz + wyz + wx'y'z' + w'xy'z'$	22				
				10	16		
		$S_2(p, q) P_1[P_2(p, q)],$ $p = P_1(w') S_2(x', z'), q = P_1(y) S_2(w', x') S_2(x, z)$				13*	13
86	83	$wxy' + wx'y + wxz + w'x'y'z'$	17				
				9	14		
		$S_2(w, p) S_2[w', S_2(z', p)] P_1[P_3(x', y', z)],$ $S_2(w, p) S_2[w', S_2(z', p)] S_3(x, y, z'),$ $p = P_2(x, y)$				12	11
87	71	$wxy + wxz + wyz + x'y'z'$	16				
				8	14		
		$S_2(z, p) P_1[P_2[p, P_3(w', x', y')] S_2(w, z)],$ $S_2(z, p) P_1[P_1(p) S_2(w, z) S_3(w, x, y)],$ $p = P_2(x, y)$				12	11
88	85	$wxy + wx'z' + w'xz$ $(w' + x' + y)(wz' + xz)$ $S_2(w', z') S_2(x', z) P_1[P_3(w', x', y)]$ $S_2(w', z') S_2(x', z) S_3(w, x, y')$	12*				
				7	11*	8	7
89	89	$wx'y + wxz' + w'xz$	12				
				7	11		
		$S_2(w', p) S_2[w, P_1(p)] S_2(x', y'), p = S_2(x, z)$				9	9
90	76	$wxz + wyz + xyz' + w'x'y'z'$	17*				
				9	14		
		$S_2(w', z) S_2(w, p) P_1[P_2(z, p) S_2(x, y)], p = P_2(x, y)$				11	11

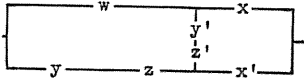
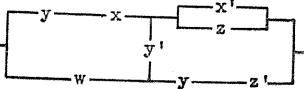
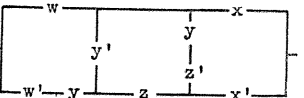
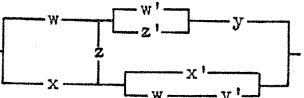
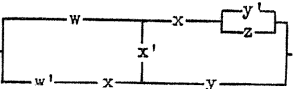
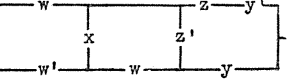


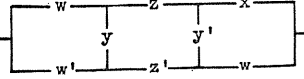
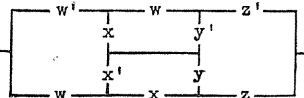
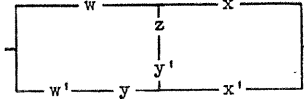
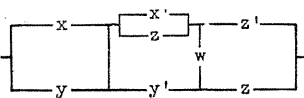
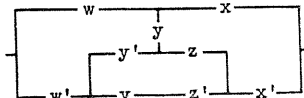
N	H	Switching Circuit	R	C	S	G	T
91	77	$wxy + wxz + wyz + w'xy'z' + w'x'yz'$	22*				
				9	14		
		$S_2(x', y') S_2(w, p) S_2[w', P_1(p)], p = P_1(z) S_2(x, y)$					10* 10
92	87	$wxz + wyz' + w'xy'z' + w'x'yz$	18				
				9	14		
		$S_2(x', y') S_2(w, p) S_2[w', P_1(p)], p = S_2(x, z) S_2(y, z')$					11 11
93	92	$wxy' + wxz' + wx'yz + w'xyz + w'x'yz'$	23				
				10	15		
		$S_2(x', y') S_2(w, p) S_2[w', P_1(p)], p = P_1(y') S_2(x, z') S_2(x', z)$					12* 12
94	78	$wxy + wxz + wyz + w'y'z'$	16				
				9	14*		
		$S_2(w', p) S_2[w, S_2(x, p) S_2(y, z)], p = S_2(y', z')$					10 10
95	97	$wx'y' + w'xy + wy'z' + xyz$	16*				
				8	12		
		$S_2(y', p) P_1[P_1(p) S_2(w, y')], p = P_1(x) S_2(w, z')$					9* 9
96	86	$wxy + wyz' + xy'z + w'x'y'z'$ $(w + y')(w'x' + y + z)(x + z')$ $S_2(w', y) S_2(x', z) P_1[S_2(w', x') P_2(y, z)]$	17*				
				8	12		
						9	9
97	94	$wxy' + w'xy + wy'z' + wx'yz$	17				
				9	14		
		$S_2(w', x') S_2(y, p) S_2[y', P_1(p)], p = P_1(w') S_2(x', z)$					10* 10

N	H	Switching Circuit	R	C	S	G	T
98	88	$wxz + wyz' + w'xy'z' + w'x'y'z$	18				
				9	15		
		$F_2[P_1(w') S_2(x', z) S_2(y', z'), P_2(w, y) S_2(x, z) S_2(x', z')]$				13	13
99	91	$wxy + wxz + wx'y'z' + w'xy'z' + w'x'yz$	23				
				11	17		
		$S_2(w, p) S_2[w', P_1(p)],$ $p = S_2(x, q) S_2[x', S_2(w, q) S_2(y, z)], q = S_2(y', z')$				15*	15
100	90	$w'x'y' + wxz + wyz'$ $w(zx + z'y') + w'x'y'$ $F_2[P_1(w') S_2(x', z) S_2(y', z'), P_3(w, x, y)]$	12				
				8	13		
						10	10
101	95	$wxy + wxz + x'y'z' + w'x'yz$	17				
				9	14		
		$S_2(x, p) P_1[P_1(p) S_2(w, x)], p = S_2(w, y) S_2(y, z') S_2(y', z)$ $S_2(x', q) P_1[P_1(x') S_2(w, q)], q = S_2(y', z') S_3(w', y, z)$				12*	
							11
102	96	$wxy + wxz + wx'y'z' + w'x'yz' + w'x'y'z$	23				
				10	17		
		$S_2[w', S_2(y, z') S_2(y', z)] S_2(x', p) S_2[x, P_1(p)],$ $p = P_1(w') S_2(y', z')$				14	14
103	104	$xyz + wxy'z' + wx'y'z' + wx'yz' + w'x'y'z'$	24				
				11	17		
		$S_2(x', p) S_2[x, P_1(p)],$ $p = S_2(w', q) P_1[P_1(q) S_2(w', x') S_2(y, z)], q = P_2(y, z)$				15*	15

N	H	Switching Circuit	R	C	S	G	T
104	105	$w'x'y + wxz + wx'y'z' + w'xy'z'$	18				
				10	15		
		$S_2(y', p) S_2(z', q) P_1[P_2(p, q) S_2(y', z')],$ $p = P_2(w, x), q = P_2(w', x')$				13*	13
105	72	$wxy + wxz + wyz + xyz + w'x'y'z'$	21				
				11	17		
		$P_2[S_2(w, y) S_2(x, y) S_2(y', z'), S_2(w', x') S_2(w, z) S_2(x, y)]$				14*	14
106	75	$wxz + wyz' + xy'z + w'x'yz$	17				
				9	14		
		$S_2(x', z') S_2(y', z') S_2(w, p) P_1[P_3(w, z', p)], p = P_2(x, y)$				12	12
107	84	$wxy + wyz' + xy'z + w'x'yz$	17				
				8	12		
		$S_2(p, q) P_1[P_2(p, q)], p = P_2(w, y'), q = S_2(x', z) S_2(y', z')$				11*	11
108	93	$wxy' + wxz' + w'yz + x'yz$	16				
				8	12		
		$S_2(p, q) P_1[P_2(p, q)], p = S_2(w, x), q = S_2(y, z)$				9	9
109	108	$wx + wy + wz$ $w(x + y + z)$ $P_2[w', P_3(x, y, z)]$ $P_1(w') S_3(x', y', z')$	9*	4	8	5	4
110	127	$wx + w'x'y + w'x'z$ $wx + w'(y + z)x'$ $S_2(w, x') S_2(w', x) P_1[P_3(w, y, z)]$ $S_2(w, x') S_2(w', x) S_3(w', y', z')$	11*	8	10	8	7

N	H	Switching Circuit	R	C	S	G	T
111	162	$wxy' + wx'y + w'xy + w'x'y'z$	17				
				9	14		
		$S_2(w', p) P_1[P_2(w) S_2(y', z'), p]],$ $p = S_2(x, y) S_2(x', y')$				12	12
112	163	$wxyz + wxy'z' + wx'y'z + wx'y'z' + w'xyz' + w'xy'z + w'x'yz$	35				
				13	20		
		$S_2(p, q) P_1[P_1(p) S_2[q, S_2(w', y')]],$ $p = S_2(w, x) S_2(w', x'), q = S_2(y, z) S_2(y', z)$				16	16
113	109	$wx + wy + xyz$	10				
				6	11		
		$S_2(x', y') S_2[w', F_3(x', y', z')]$				7	7
114	110	$wx + wy + xy'z$ $(w + zy')(x + y)$ $S_2(w', y) S_2(w', z') S_2(x', y')$	10*		5	9	6
115	112	$wx + wyz + wy'z' + xyz$	15				
				8	14		
		$F_2[P_1(w') S_2(x', y) S_2(x', z), S_2(w', x') P_2(y', z')]$				11	11
116	113	$wx + wyz' + wy'z + xyz$	15*				
				7	12		
		$P_2[P_1(w) S_2(y, z), P_1(x) S_2(y, z') S_2(y', z)]$				10*	10
117	111	$wx + wy + w'x'y'z$	11				
				7	11		
		$S_2(w, p) S_2[w', S_2(z, p)], p = P_2(x, y)$				8	8

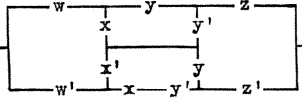
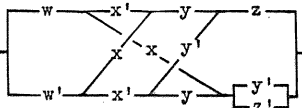
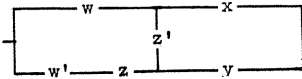
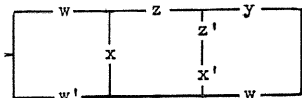
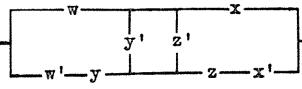
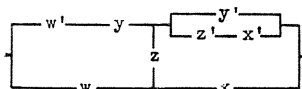
N	H	Switching Circuit	R	C	S	G	T
118	114	$wx + wy'z' + x'yz$	11				
				7	12		
		$S_2(w', x) S_2(w', y') S_2[x', S_2(y, z) S_2(y', z')]$ $P_1[S_2(w, x) S_3(w, y', z') S_3(x', y, z)]$				10	9
119	136	$wx'y' + wyz' + wy'z + xyz$	16*				
				8	13*		
		$F_2[P_2(w', y) S_2(x, z'), P_1(y') S_2(w', z') S_2(x', z)]$ $S_2(w', p) S_2[x', P_1(p)] S_3(x, y', z'), p = S_2(y, z)$				11	10
120	115	$wx + wyz' + wy'z + w'x'yz$	16				
				9	14		
		$S_2(w, p) P_1[P_1(w) S_2(y, p)], p = P_1(x) S_2(y, z') S_2(y', z)$				11*	11
121	146	$wxy' + wx'z + wyz' + w'xyz$	17				
				9	14		
		$S_2(w, p) S_2[w', P_1(p)] P_1[P_3(x, y, z)],$ $S_2(w, p) S_2[w', P_1(p)] S_3(x', y', z'),$ $p = P_3(x', y', z')$				12	11
122	137	$wxy' + wx'y + w'xy + wxz$	16*				
				8	13		
		$S_2(w', p) P_1[P_3(w', z, p)] S_2(x', y'), p = S_2(x, y)$				10*	10
123	123	$wy + w'xy' + xy'z$ $wy + (w' + z)xy'$ $S_2(w', y) S_2(x', y') P_1[P_3(w', y, z)]$ $S_2(w', y) S_2(x', y') S_3(w, y', z')$	11*				
				6	10	8	7
124	140	$wxy + wyz' + wy'z + xyz' + xy'z$	20*				
				8	13		
		$S_2(w', x') S_2(y', z') P_1[S_2(w, x) P_2(y', z')]$				9	9

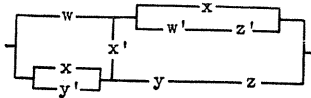
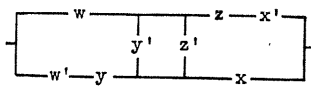
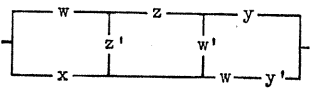
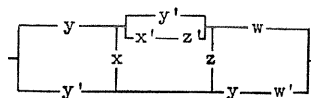
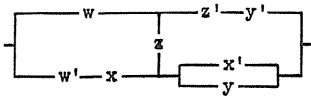
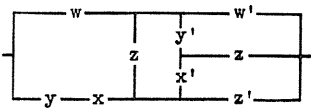
N	H	Switching Circuit	R	C	S	G	T
125	130	$wyz' + wy'z + xyz + w'xy'z'$	17				
				8	15		
		$S_2(w', p) P_1[P_2[P_1(x') S_2(w, y'), p]],$ $p = S_2(y, z) S_2(y', z')$				12*	12
126	128	$wxy + wxz + wyz + xyz + wx'y'z' + w'xy'z'$	26*				
				10	16		
		$S_2(w', x') S_2(p, q) P_1[P_2(p, q) S_2(y, z)],$ $p = P_2(w', x'), q = P_2(y, z)$				13*	13
127	126	$wx + w'x'y + wy'z$	11				
				7	11		
		$S_2(w', x) S_2(w', y') P_1[P_2(w', x) S_2(y', z)]$				9	9
128	124	$wx + x'yz + w'x'y'z'$ $xw + x'(y' + z)(y + z'w')$ $S_2(w', x) S_2[x', S_2(y, z) P_1[P_3(w, y, z)]]$ $S_2(w', x) S_2[x', S_2(y, z) S_3(w', y', z')]$	12				
				8	12	10	9
129	152	$wxz' + wyz' + xy'z + x'yz$ $(w + z)(x' + y' + z')(x + y)$ $S_2(w', z') S_2(x', y') P_1[P_3(x', y', z')]$ $S_2(w', z') S_2(x', y') S_3(x, y, z)$	16*				
				7	11	8	7
130	142	$wxy' + wyz + xy'z + x'yz'$	16*				
				8	13		
		$S_2(x', y') P_2[P_2(x', z) S_2(w, y'), P_3(w, y', z')]$ $S_2(x', y') P_1[P_2(x', z) S_2(w, y')] S_3(w', y, z)$				11	10
131	125	$wx + wyz + w'x'yz' + w'x'y'z$	17				
				9	14		
		$S_2(w, p) S_2[w', S_2[p, S_2(y', z')]], p = P_1(x) S_2(y, z)$				11*	11

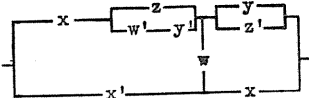
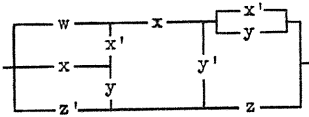
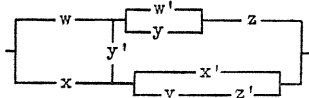
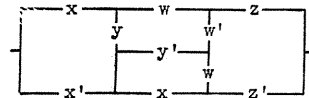
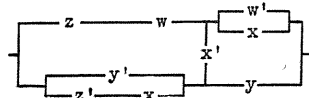
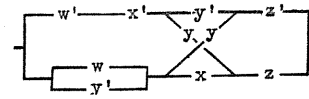
N	H	Switching Circuit	R	C	S	G	T
132	149	$wxy' + wx'y + w'xy + wx'z$	16*				
				8	13		
		$S_2(w', p) S_2[w, P_1(p)] P_1[P_3(x, y, z)], p = S_2(x, y)$				11	
		$S_2(w', p) S_2[w, P_1(p)] S_3(x', y', z'), p = S_2(x, y)$					10
133	133	$wx'y + wx'z + xyz + xy'z'$	16*				
				8	13		
		$F_2[P_1(x') S_2(y, z') S_2(y', z), P_2(w', x) S_2(y', z')]$				11	11
134	139	$wxy' + wx'y + xyz + w'x'y'z'$	17				
				10	15*		
		$P_2[S_2(w, x') S_2(x, z) S_2(y', z'), P_1(y) S_2(w, x) S_2(w', x')]$				13*	13
135	151	$wxy' + wx'y + wxz' + w'xyz + w'x'y'z$	22*				
				10	15		
		$S_2(w', p) P_1[P_1(p) S_2(w', z)], p = S_2(x', y') P_1[P_3(x', y', z')]$				12	
		$S_2(w', q) P_1[P_1(q) S_2(w', z)], q = S_2(x', y') S_3(x, y, z)$					11
136	147	$wxy' + wx'y + wxz + w'xyz' + w'x'y'z$	22*				
				10	15		
		$S_2(w, p) S_2[w', S_2[p, S_2(y', z')]],$				14*	
		$p = S_2(x, y') S_2(x', y) S_2(x, z)$					
		$S_2(w', q) P_1[P_2[P_1(w) S_2(y', z'), q]],$					13
		$q = S_2(x', y') S_3(x, y, z)$					
137	134	$wx'y + wx'z + wyz + wxy'z' + w'xyz' + w'xy'z$	27*				
				11	17		
		$S_2(w, p) P_1[P_2[P_1(w') S_2(x', q), p]],$				14*	14
		$p = P_2(x', q) S_2(y, z), q = P_2(y, z)$					

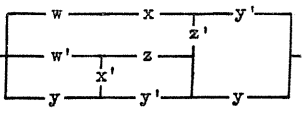
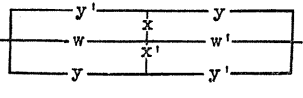
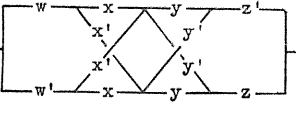
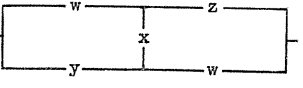
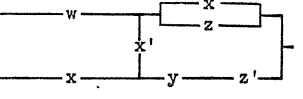
N	H	Switching Circuit	R	C	S	G	T
138	155	$wx'y + w'xy' + wx'z' + xyz$ $(w+x)[(w'+y)(y'+z) + x'z']$ $S_2(w', x') P_1[S_2(w', y') S_2(x', z') S_2(y, z)]$	16*				
				8	12		9* 9
139	145	$wx'y + w'xy' + wx'z + w'xz' + wyz$	20*				
				10	15*		
		$S_2(w, p) P_1[P_2[P_1(w') S_2(y', z'), p]], p = P_1(x') S_2(y, z)$					11* 11
140	157	$w'xy' + wx'z + w'xz + wyz'$	16*				
				8	13		
		$S_2(w, p) S_2[w', S_2(x, p)], p = S_2(x', z) S_2(y, z')$					10 10
141	161	$wx'y + w'xy + wx'z + w'xz + wxy'z'$	21*				
				9	14		
		$S_2(p, q) P_1[P_2(p, q)],$ $p = S_2(w', x') S_2(y', z'), q = P_2(w', x')$					11* 11
142	156	$wxy + w'x'y + wy'z + w'xy'z'$	17				
				9	14		
		$S_2(w, p) S_2[w', S_2[p, S_2(x', y')]], p = S_2(x, y) S_2(y', z)$					12 12
143	158	$wxy + w'x'y + wxz' + wx'y'z + w'xy'z$	22				
				10	18		
				11	17		
		$S_2(w, p) P_1[P_2[p, P_1(w') S_2(x', z')]],$ $p = S_2(x, y) S_2(x', y') S_2(x, z')$ $S_2(w', q) S_2[w, S_2[q, S_2(x', z')]], q = S_2(x', y) S_3(x, y', z)$					14
							13

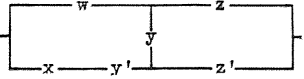
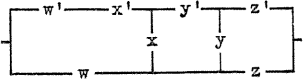
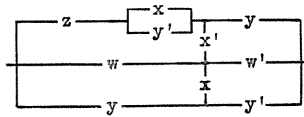
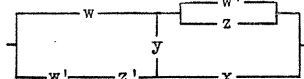
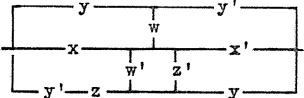


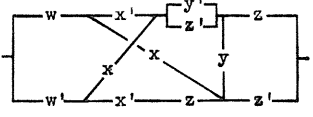
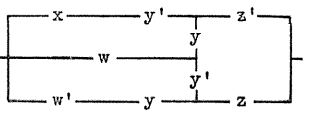
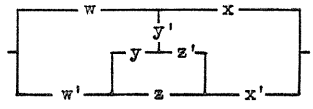
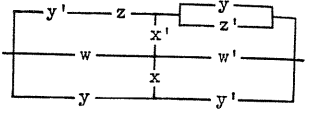
N	H	Switching Circuit	R	C	S	G	T
144	144	$wxy + wxz + wyz + w'xy'z' + w'x'yz' + w'x'y'z$ 	27				
		$S_2(w', p) S_2[w, P_1(p) S_2(x', q),$ $p = S_2(x', y') S_2(x', z') P_1(q), q = P_2(y, z)$		11	17		
						14	14
145	160	$wxy' + wxz' + wx'yz + w'xyz + w'x'yz' + w'x'y'z$ 	28				
		$S_2(p, q) P_1[P_2(p, q)],$ $p = S_2(y, z), q = S_2(w, x) P_1[P_2(w, x) S_2(y', z')]$		12	18		
						14*	14
146	116	$wx + yz$ $P_1[S_2(w, x) S_2(y, z)]$	6	4	8		
						5	5
147	119	$wx + wyz' + w'yz$ 	11				
		$F_2[P_2(w', x'), P_1(y') S_2(w, z) S_2(w', z')]$		6	10		
						9	9
148	117	$wx + wyz + xyz + w'x'yz'$ 	16				
		$S_2(z, p) P_1[S_2(w, x) S_2[y, F_2(z, p)], p = P_2(w, x)]$		8	13		
						11*	11
149	121	$wx + wy'z' + w'yz$ $w(x + y'z') + w'yz$ $P_2[P_1(w) S_2(y, z), P_2(w', x) S_2(y', z')]$	11				
				7	12		
						9	9
150	122	$wx + wy'z + xyz' + w'x'yz$ 	16*				
		$S_2(w', y') S_2(x', z') P_1[P_2(y', z') S_2(w, x) S_2(w', x')]$		8	12*		
						11*	11
151	120	$wx + wy'z + xyz + w'x'yz'$ 	16*				
		$P_2[P_1(w) S_2(x', y) S_2(y, z), P_1(x) S_2(w', z') S_2(y', z)]$ $S_2(w', y') S_3(w, x', z') S_3(w', x, z') S_3(x', y, z)$		8	12		
						12	
							11

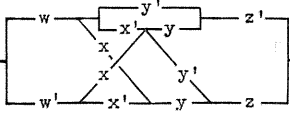
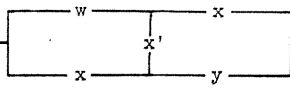
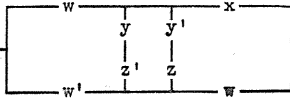
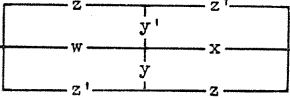
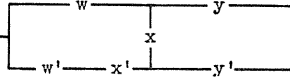
N	H	Switching Circuit	R	C	S	G	T
152	118	$wx + wyz + xyz + w'x'y'z'$	16				
				9	14		
		$P_2[S_2(w, x) S_2(w', x') S_2(y, z), P_2(w, x), S_2(y', z')]$				12*	12
153	150	$wxy' + w'xy + wxz' + wx'z$	16*				
				8	12		
		$S_2(x', z') S_2(w', p) P_1[P_3(w', z', p)], p = S_2(x, y)$				10	10
154	138	$wxy' + w'xy + wyz + wy'z'$	16*				
				8	13		
		$F_2[P_1(y') S_2(w', x') S_2(w, z'), P_2(w', y) S_2(x', z)]$ $S_2(w', x') S_2(w', y') S_3(w, y, z') S_3(x', y', z)$				11*	10
155	131	$wxy' + w'xy + wy'z + xyz + wx'y'z'$	21*				
				10	15		
		$S_2(p, q) P_1[P_2(p, q)],$ $p = S_2(w', y') S_2(x', y), q = P_2(w', z) S_2(x, y')$				13*	13
156	153	$wx'y' + w'xy + wyz + wy'z'$	16*				
				8	12*		
		$S_2(w', x') S_2(y, p) P_1[P_3(x', y, p)], p = P_2(w', z)$				10*	10
157	135	$wx'y' + w'xy + wx'z + wy'z + xyz'$	20*				
				9	15*		
		$S_2(p, q) P_1[P_2(p, q)],$ $p = P_2(x', y'), q = P_1(w') S_2(x, z') S_2(y, z')$				12*	12

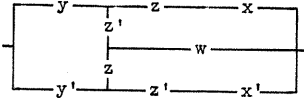
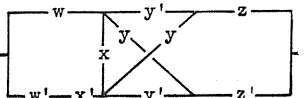
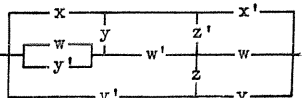
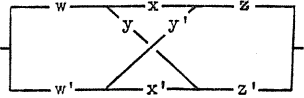
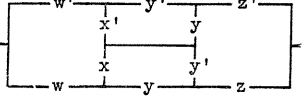
N	H	Switching Circuit	R	C	S	G	T
158	141	$wxz + wx'z' + wyz + xyz + w'xy'z'$	21*				
				9	15		
		$S_2(w', x') S_2(z, p) P_1[P_3(x', z, p)], p = P_1(y) S_2(w, x)$				11*	11
159	129	$wxy + wxz + wyz + xyz + x'y'z'$	20				
				10	16		
		$F_3[P_2(w', x') S_2(y', z'), S_2(w', x') P_2(y', z'), P_3(x, y, z)]$ $S_3(x, y', z') P_1[S_2(w, x) S_2(y', z') P_1[S_2(w', x') P_2(y', z')]]$				14	13
160	148	$wx'y' + wyz + xyz' + w'xy'z$	17				
				9	14*		
		$S_2(w', x') S_2(y', p) P_1[P_2(y', p) S_2(w', z)], p = P_1(x) S_2(w', z)$				12*	12
161	132	$wxz + wyz + xyz' + wx'y'z' + w'x'y'z$	22				
				10	16		
		$S_2(p, q) P_1[P_2(p, q)],$ $p = P_2(x, y) S_2(w', z'), q = S_2(w', z) S_2(x', z') S_2(y', z')$				15*	15
162	143	$w'x'y' + wxz + wyz + xyz'$	16				
				9	14		
		$S_2(w, p) P_1[S_2(w, z) P_2[p, P_3(x', y', z)]], p = P_2(x, y)$ $S_2(w, p) P_1[P_1(p) S_2(w, z) S_3(x, y, z')], p = P_2(x, y)$				12*	11
163	154	$wxy + wyz' + xy'z + w'x'yz + w'x'y'z'$	22				
				10	16*		
		$S_2(p, q) P_1[P_2(p, q)],$ $p = P_1(w) S_2(x, y') S_2(y', z'), q = S_2(x', z) S_2(y', z')$				14*	14

N	H	Switching Circuit	R	C	S	G	T
164	159	$wxy' + wxz' + w'yz + x'yz + w'x'y'z'$	21				
				10	16		
		$S_2(p, q) P_1[P_2(p, q) P_1[P_4(w, x, y, z)]]$ , $S_2(p, q) P_1[P_2(p, q) S_4(w', x', y', z')]$ , $p = P_2(w', x')$ , $q = P_2(y', z')$			14*		13
165	164	$w$	0	1	2	0	0
166	204	$wx' + w'x$ $S_2(w, x) S_2(w', x')$ $P_2(w, x') + P_2(w', x)$	6	4	6		
						4	4
							4
167	236	$wxy + wx'y' + w'xy' + w'x'y$	16				
				8	12		
		$S_2(w, p) S_2[w', P_1(p)]$ , $p = S_2(x, y) S_2(x', y')$ $P_2(w, q) + P_2(w', S_1(q))$ , $q = P_2(x, y) + P_2(x', y')$				9	9
							9
168	237	$wxyz' + wxy'z + wx'y'z + wx'y'z' + w'xyz + w'xy'z' + w'x'yz' + w'x'y'z$	40				
				12	18		
		$S_2(p, q) P_1[P_2(p, q)]$ , $p = S_2(w, x) S_2(w', x')$ , $q = S_2(y, z) S_2(y', z')$ $P_2(r, s) + S_1[S_2(r, s)]$ , $r = P_2(w, x') + P_2(w', x)$ , $s = P_2(y, z) + P_2(y', z')$				13*	13
							13
169	165	$wx + wy + wz + xyz$	13				
				5	10		
		$S_2[w', F_3(x', y', z')] P_1[P_3(x, y, z)]$ $S_2[w', S_3(x, y, z)] S_3(x', y', z')$ $P_2[w', P_3(x, y, z)] + P_3(x', y', z')$				9	
							8
							8
170	166	$wx + wy + wz + xyz'$	13				
				7	12		
		$S_2[w', F_3(x', y', z)] P_1[P_3(x, y, z)]$ $S_2[w', S_3(x, y, z')] S_3(x', y', z')$ $P_2[w', P_3(x, y, z)] + P_3(x', y', z)$				9	
							8
							8

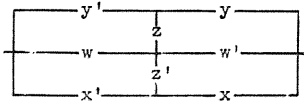
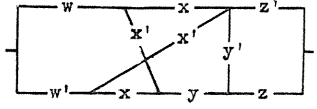
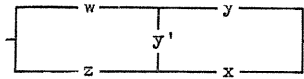
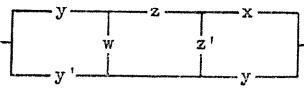
N	H	Switching Circuit	R	C	S	G	T
171	167	$wy + wz + xy'z'$	10				
				6	10		
		$S_2(w', p) S_2[x', P_1(p)], p = S_2(y', z')$ $P_2(w', q) + P_2[x', S_1(q)], q = P_2(y, z)$				7*	7
172	168	$wx + wy + wz + w'x'y'z'$	14				
				8	12		
		$S_2(w, p) S_2[w', P_1(p)], p = P_3(x, y, z)$ $P_2(w, q) + P_2[w', S_1(q)], q = S_3(x', y', z')$				8	8
173	208	$wxy' + wx'y + w'xy + wxz + w'x'y'z$	21				
				10	16		
		$S_2(w', p) S_2(z', q) P_1[P_2(w', p) S_2(y, z)],$ $p = P_1(q) S_2(x, y), q = P_2(x, y)$ $P_2(w', r) + P_2(z', s) + S_1[S_2(w', r) + P_2(y, z)],$ $r = S_1(s) + P_2(x, y), s = S_2(x, y)$				14*	14
174	201	$wz + w'xz' + w'yz' + xyz'$	15				
				7	13		
		$S_2(w', z) P_1[P_1(z) S_2(w', x) S_2(w', y) S_2(x, y)]$ $P_2(w', z') + S_1[S_1(z') + P_2(w', x) + P_2(w', y) + P_2(x, y)]$				10*	10
175	220	$wxy' + wx'y + w'xy + wxz' + w'x'y'z$	21				
				10	15		
		$S_2(w', p) S_2(z', q) P_1[P_2(w', p) S_2(x, z')],$ $p = P_1(q) S_2(x, y), q = P_2(x, y)$ $P_2(w', r) + P_2(z, s) + S_1[S_2(w', r) + P_2(x, z)],$ $r = S_1(s) + P_2(x, y), s = S_2(x, y)$				14*	14

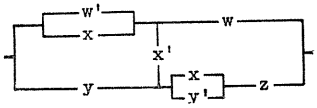
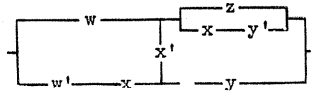
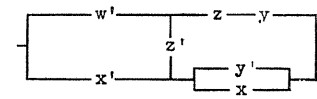
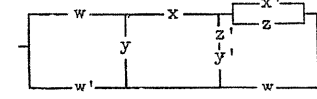
N	H	Switching Circuit	R	C	S	G	T
176	205	$wxy + wxz' + wyz' + xyz' + wx'y'z + w'xy'z + w'x'yz$	31				
				12	18*		
		$S_2(p, q) P_1[P_2(p, q)], p = P_1(r) S_2(y', z)$				16*	16
		$q = S_2(w, x) S_2(y, z') S_2(y', r), r = P_2(w, x)$					
		$P_2(s, t) + S_1[S_2(s, t)], s = S_1(u) + P_2(y', z')$					16
		$t = P_2(w, x) + P_2(y, z) + P_2(y', u), u = S_2(w, x)$					
177	202	$wy + w'y'z + xy'z'$ $yw + y'(zw' + z'x)$ $S_2(w', y) P_1[P_1(y) S_2(w', z) S_2(x, z')]$ $P_2(w', y') + S_1[S_1(y') + P_2(w', z') + P_2(x, z)]$	11				
				7	11	8*	8
							8
178	210	$wyz' + wy'z + w'yz + xy'z'$	16				
				9	14		
		$S_2(w, p) S_2(x', q) P_1[P_3(w, p, q)],$				12	12
		$p = P_2(y', z'), q = P_2(y, z)$					
		$P_2(w, r) + P_2(x', s) + S_1[S_3(w, r, s)],$					12
		$r = S_2(y, z), s = S_2(y', z')$					
179	203	$wx + w'x'y + w'x'z + wy'z'$	15				
				8	12		
		$S_2(w', p) S_2[w, P_1(p)], p = P_1(x) S_2(y', z')$				8	8
		$P_2(w', q) + P_2[w, S_1(q)], q = S_1(x') + P_2(y, z)$					8
180	228	$wxy' + wx'y + w'xy + wx'z' + w'x'y'z$	21				
				10	15		
		$S_2(w, p) S_2[w', P_1(p)], p = S_2(x, y') S_2(x', y) S_2(y', z')$				11	
		$S_2(w', q) S_2[w, P_1(q)], q = S_2(x, y) S_3(x', y', z)$					10
		$P_2(w', r) + P_2[w, S_1(r)], r = P_2(x', y') + P_3(x, y, z')$					10

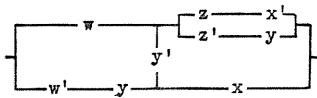
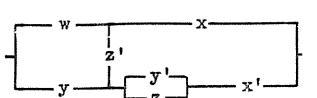
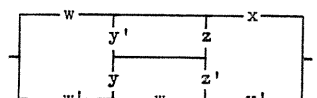
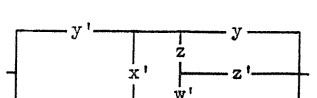
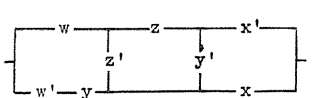
N	H	Switching Circuit	R	C	S	G	T
181	215	$wx'y' + wx'z' + wy'z' + wxyz + w'xyz' + w'xy'z + w'x'yz$ 	32		12	18	
		$S_2(w, p) S_2[w', P_1(p)],$ $p = S_2(x, q) S_2[x', P_1(q)] S_2(y', z'), q = P_2(y', z')$ $P_2(w, r) + P_2[w', S_1(r)],$ $r = P_2(x', s) + P_2[x, S_1(s)] + P_2(y, z), s = S_2(y, z)$				14	14
182	169	$wx + wy + xy$ 	9		5	9	
		$S_2(w', x') S_2(w', y') S_2(x', y')$ $P_2(P_2(w', x') + P_2(w', y') + P_2(x', y'))$				6	6
183	174	$wy' + xy$ $S_2(w', y') S_2(x', y)$ $(w', y) + P_2(x', y')$	6	4	7	4	4
184	180	$wx + wyz' + wy'z + xyz' + xy'z$ 	19		8	13	
		$S_2(w', x') P_1[S_2(w, x) S_2(y, z') S_2(y', z)]$ $P_2(w', x') + S_1[P_2(w, x) + P_2(y, z) + P_2(y', z')]$				9*	9
185	184	$wyz + wy'z' + xyz' + xy'z$ 	16		8	13	
		$S_2(w', p) S_2[x', P_1(p)], p = S_2(y, z') S_2(y', z)$ $P_2(w', q) P_2[x', S_1(q)], q = P_2(y, z') + P_2(y', z)$				9	9
186	179	$wx + wy + w'x'y'$ 	10		6	9	
		$S_2(w, p) S_2[w', P_1(p)], p = P_2(x, y)$ $P_2(w, q) + P_2[w', S_1(q)], q = S_2(x', y')$				7	7

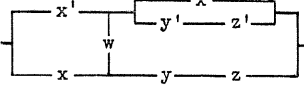
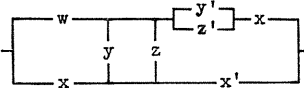
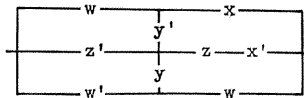
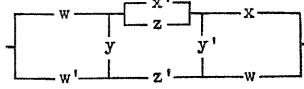
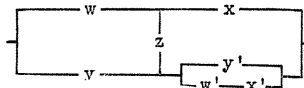
N	H	Switching Circuit	R	C	S	G	T
187	189	$wx + x'yz + x'y'z'$ $xw + x'(yz + y'z')$ $S_2(w', x) S_2[x', S_2(y, z) S_2(y', z')]$ $P_2(w', x') + P_2[x, P_2(y, z') + P_2(y', z)]$	11		7 11	8 8	8
188	219	$wyz' + wy'z + xyz + x'y'z'$  $S_2(x, p) S_2(x', q) P_1[P_3(w, p, q)],$ $p = P_2(y, z), q = P_2(y', z')$ $P_2(x, r) + P_2(x', s) + S_1[S_3(w, r, s)],$ $r = S_2(y', z'), s = S_2(y, z)$	16		9 14	12 12	12
189	190	$wx + wyz' + wy'z + w'x'yz + w'x'y'z'$  $S_2(w, p) S_2[w', P_1(p)], p = P_1(x) S_2(y, z') S_2(y', z)$ $P_2(w, q) + P_2[w', S_1(q)], q = S_1(x') + P_2(y, z') + P_2(y', z)$	21		10 15	10 10	10
190	226	$wxy' + wx'z + wyz' + w'xyz + w'x'y'z'$  $S_2(w, p) S_2[w', P_1(p)], p = S_2(x, y') S_2(x', z) S_2(y, z')$ $P_2(w, q) + P_2[w', S_1(q)], q = P_2(x', y) + P_2(x, z') + P_2(y', z)$	22		11 17	11 11	11
191	233	$wxz + w'x'z' + wyz' + w'y'z$  $F_2[S_2(w, x') S_2(w', y) S_2(y', z'), P_2(x, z) S_2(w, y')]$ $B_2[P_2(w, x) + P_2(w', y') + P_2(y, z'), S_2(x', z) + P_2(w, y)]$	16		8 14	12 12	12
192	225	$wxy + w'x'y' + wxz + w'x'z' + wyz + w'y'z'$  $S_2(w, p) S_2[w', P_1(p)], p = S_2(x, y) S_2(x, z) S_2(y, z)$ $P_2(w, q) + P_2[w', S_1(q)], q = P_2(x', y') + P_2(x', z') + P_2(y', z')$	24		10 15	11 11	11

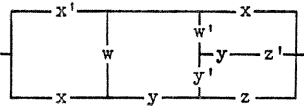
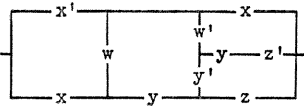
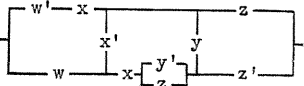
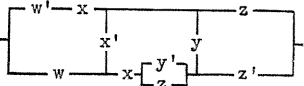
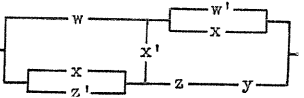
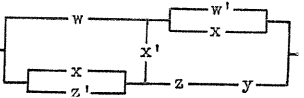


N	H	Switching Circuit	R	C	S	G	T
193	234	$wxz' + w'x'z' + wyz + w'y'z$	16				
				8	12		
		$S_2(w, p) S_2[w', P_1(p)], p = S_2(x, z') S_2(y, z)$				9	9
		$P_2(w, q) + P_2[w', S_1(q)], q = P_2(x', z) + P_2(y', z')$					9
194	235	$wxy' + w'x'y' + wxz' + w'x'z' + wx'yz + w'xyz$	26				
				10	15		
		$S_2(p, q) P_1[P_2(p, q)],$				11*	11
		$p = S_2(w, x) S_2(w', x'), q = S_2(y, z)$					
		$P_2(r, s) + S_1[S_2(r, s)],$					11
		$r = P_2(w, x') + P_2(w', x), s = P_2(y', z')$					
195	170	$wx + wy + xz$	9				
				5	9		
		$S_2(w', x') S_2(w', z') S_2(x', y')$				6	6
		$P_2(w', x') + P_2(w', y') + P_2(x', z')$					6
196	171	$wx + wy + xyz + xy'z'$	14				
				7	12		
		$F_2[P_1(w') S_2(x', y'), P_1(x') S_2(y, z') S_2(y', z)]$				10	10
		$B_2[S_1(x') + P_2(w', y'), S_1(w') + P_2(y, z) + P_2(y', z')]$					10
197 a	175	$wx + wy + xy'z + x'yz$	14*				
		$[w + z(x' + y')](x + y)$		6	10		
		$S_2(w', z') S_2(x', y') P_1[P_3(w, x', y')]$				8	
		$S_2(w', z') S_2(x', y') S_3(w', x, y)$					7
		$S_1[P_2(w, z) + P_2(x, y) + P_3(w, x', y')]$					8
197 b	181	$wz + xy + wx'y'$	10				
		$w(z + x'y') + xy$		6	10		
		$P_1[S_2(w, z) S_2(x, y) P_1[P_3(w', x, y)]]$				9	
		$P_1[S_2(w, z) S_2(x, y) S_3(w, x', y')]$					8
		$P_2(w', z') + P_2(x', y') + P_3(w', x, y)$					7

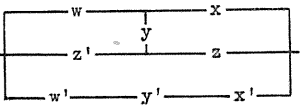
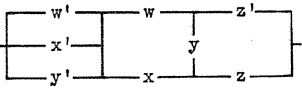
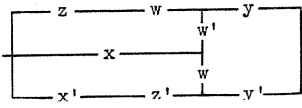
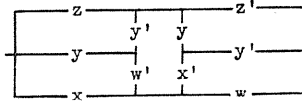
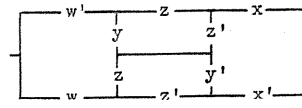
N	H	Switching Circuit	R	C	S	G	T
198 a	172	$wx+wy+xyz+w'x'y'z$	15				
				8	13		
		$S_2(w, p) S_2(w', z') P_1[P_2(w, p) S_2(x, y)], p=P_2(x, y)$				11	11
		$P_2(w', q)+S_1[S_2(w', z)+P_2[q, P_2(x', y')]], q=P_2(x, y)$					11
198 b	186	$wz+wx'y'+wx'y+w'xy$	15				
				8	13		
		$S_2(w', p) P_1[P_2(w', z) S_2[p, S_2(x', y')]], p=S_2(x, y)$				11*	11
		$P_2(w, q)+P_2(w', z')+S_1[S_2(w, q)+P_2(x, y)], q=S_2(x, y)$					11
199	177	$wx+wz+w'yz'$ $w(x+z)+w'yz'$ $S_2(w', y') S_2(w', z) P_1[P_3(w', x, z)]$ $S_2((w', y') S_2(w', z) S_3(w, x', z'))$ $P_2(w', x')+P_2(w', z')+P_3(w, y', z)$	10	6	10	8	7
							7
200 a	176	$wx+wy+xy'z+x'y'z'$ $(x+y)(w+x'z'+y'z)$ $S_2(x', y') P_1[P_1(w) S_2(x', z') S_2(y', z)]$ $B_2[S_2(x', y'), S_1(w')+P_2(x, z)+P_2(y, z')]$	14*	7	11	8	8
							9
200 b	194	$xy+wx'z+wy'z'$	11				
				7	11*		
		$F_2[P_2(x', y'), P_1(w') S_2(x, z) S_2(y, z')]$				9*	9
		$P_2(x', y') S_1[S_1(w)+P_2(x', z')+P_2(y', z)]$					8
201 a	173	$wx+wy+xyz+w'x'y'z'$	15				
				9	14		
		$P_2[P_1(w) S_2(x, z) S_2(y', z'), P_1(y) S_2(w, x) S_2(w', x')]$				12	
		$S_2[w', S_3(x, y, z) S_3(x', y', z)] S_3(w, x', y')$					11
		$P_2(w', p)+S_1[S_3(w', z', p)]+P_3(x', y', z'), p=P_2(x, y)$					11

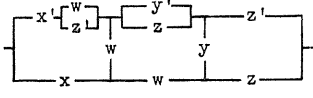
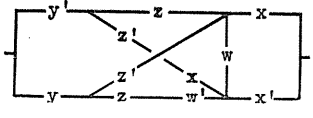
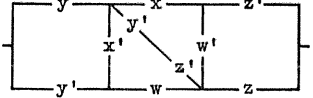
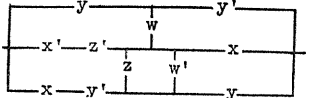
N	H	Switching Circuit	R	C	S	G	T
201 b	217	$wx'y + w'xy + wxz' + wy'z$	16*				
				9	14		
		$F_2[P_1(w') S_2(x', z') S_2(y, z), P_1(y') S_2(w, x) S_2(w', x')]$				12	
		$S_2(w', p) P_1[P_3(w', z', p)] S_3(x', y', z'), p = S_2(x, y)$					11
		$P_2[w', P_3(x, y, z) + P_3(x', y', z') + P_3(w, x', y')]$					11
202	185	$wx + wy'z + xyz' + x'yz$	15				
				7	11		
		$S_2(w', y') S_2(z', p) P_1[P_2(z', p) S_2(w, x)], p = P_2(x, y')$				11*	11
		$P_2(w', x') + P_2(z', q) + S_1[S_2(z', q) + P_2(w, y)], q = S_2(x', y)$					11
203	182	$wx + wyz + wy'z' + xyz + w'x'yz'$	20				
				9	14		
		$S_2(w', y') S_2(w, p) P_1[P_2(w, p) S_2(x, z)],$				14*	14
		$p = P_1(x) S_2(y, z) S_2(y', z')$					
		$P_2(w', x') + P_2(w, q) + S_1[S_2(w, q) + P_2(y, z')],$					14
		$q = S_1(y) + P_2(x, z') + P_2(x', z)$					
204 a	178	$wx + wy + xy'z + w'x'y'z'$	15				
				8	12		
		$S_2(p, q) P_1[P_2(p, q)], p = S_2(x', y'), q = P_1(w) S_2(y', z)$				10*	10
		$P_2(r, s) + S_1[S_2(r, s)], r = S_2(x', y'), s = S_1(w') + P_2(y, z')$					10
204 b	188	$wy' + w'xy + wx'z + xyz'$	15				
				8	12		
		$S_2(p, q) P_1(P_2(p, q)), p = P_2(x', y'), q = P_1(w') S_2(y, z')$				10*	10
		$P_2(r, s) + S_1[S_2(r, s)], r = P_2(x', y'), s = S_1(w) + P_2(y', z)$					10

N	H	Switching Circuit	R	C	S	G	T
205 a	183	$wx + wyz + xyz + x'y'z'$	15				
				8	14		
		$F_3[P_2(w', x'), S_2(w', x') P_2(y', z'), P_3(x, y, z)]$					12
		$P_1[S_2(w, x) P_1[S_2(w', x') P_2(y', z')] S_3(x', y', z')]$					11
		$P_2(w', x') + S_1[P_2(w, x) + S_2(y, z)] + P_3(x, y, z)$					10
205 b	218	$wx'y + wxz' + wy'z + xyz' + xy'z$	20*				
				8	13		
		$S_2(w', x') P_2[S_2(w, x) P_2(y, z), P_3(x', y', z')]$					11
		$S_2(w', x') P_1[S_2(w, x) P_2(y, z)] S_3(x, y, z)$					10
		$S_1[P_2(w, x) + S_1[P_2(w', x') + S_2(y', z')]] + P_3(x', y', z')$					11
206 a	187	$wx + wyz' + wy'z + xy'z' + w'x'yz$	20				
				9	14		
		$S_2(w, p) S_2(x', q) P_1[P_3(w, p, q)],$					13* 13
		$p = P_3(x, y', z'), q = P_2(y, z)$					
		$P_2(w, r) + P_2(x', s) + S_1[S_3(w, r, s)],$					13
		$r = S_3(x', y, z), s = S_2(y', z')$					
206 b	207	$wx'y' + wxz + wyz' + xyz + w'xy'z'$	21*				
				9	16		
		$S_2(w, p) S_2(x', q) P_1[P_3(w, p, q)],$					13* 13
		$p = P_3(x', y, z), q = P_2(y', z')$					
		$P_2(w, r) + P_2(x', q) + S_1[S_3(w, r, s)],$					13
		$r = S_3(x, y', z'), s = S_2(y, z)$					
207	195	$wx + w'x'y + wy'z + w'yz$	15				
				7	12		
		$F_2[P_2(w, y') S_2(x, z'), P_1(w') S_2(x', y) S_2(x', z')]$					11*
		$S_2(w', y') P_1[P_2(w', x) S_2(y', z)] S_3(w', x, z')$					10
		$P_2(w', x') + S_1[S_2(w', y) + P_2(x', z)] + P_3(w', y, z')$					10

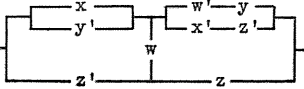
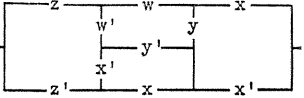
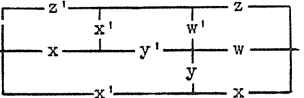
N	H	Switching Circuit	R	C	S	G	T
208 a	191	$wx+yz+w'x'y'z$ $wx+(y+w'z')(z+y'x')$ $P_1[S_2(w, x)S_2(y, z)P_1[P_4(w, x, y, z)]]$ $P_1[S_2(w, x)S_2(y, z)S_4(w', x', y', z')]$ $P_2(w', x')+P_2(y', z')+P_4(w, x, y, z)$	11		8 12	10	9 8
208 b	231	$w'xy+wx'z+wy'z'+xy'z$ $(w+x)(w'+x'+y'+z')(y+z)$ $S_2(w', x')S_2(y', z')P_1[P_4(w', x', y', z')]$ $S_2(w', x')S_2(y', z')S_4(w, x, y, z)$ $S_1[P_2(w, x)+P_2(y, z)+P_4(w', x', y', z')]$	16*		8 12	9	8 9
209 a	192	$wx+wyz+xyz+w'x'yz'+w'x'y'z$  $S_2[p, P_1(q)]S_2[q, P_1(p)],$ $p=S_2(w, x)S_2(y, z), q=P_2(w, x)S_2(y', z')$ $P_2(r, s)+S_1[S_2(r, s)],$ $r=P_2(w', x')+P_2(y', z'), s=S_2(w', x')+P_2(y, z)$	21				
				10	16		
		$S_2(p, q)P_1[P_2(p, q)],$ $p=S_2(w', x')S_2(y', z'), q=P_2(w', x')S_2(y, z)$ $P_2[r, S_1(s)]+P_2[s, S_1(r)],$ $r=P_2(w, x)+P_2(y, z), s=S_2(w, x)+P_2(y', z')$				14* 14	13
209 b	211	$wx'y+w'xy+wx'z+w'xz+wy'z'+wxy'z'$ 	25*				
				10	16		
		$S_2(p, q)P_1[P_2(p, q)],$ $p=S_2(w', x')S_2(y', z'), q=P_2(w', x')S_2(y, z)$ $P_2[r, S_1(s)]+P_2[s, S_1(r)],$ $r=P_2(w, x)+P_2(y, z), s=S_2(w, x)+P_2(y', z')$				13 13	14
210	198	$wx+w'x'y+wy'z'+w'yz$ $[w+y(x'+z)](w'+x+z'y')$ $F_2[P_2(w, y')S_2(x, z'), P_1(w')S_2(x', y)S_2(x', z)]$ $S_2(w', y')P_1[P_2(w', x)S_2(y', z')]S_3(w', x, z')$ $P_2(w', x')+S_1[S_2(w', y)+P_2(x', z)]+P_3(w', y, z)$	15		8 13*	11	10 10
211 a	193	$wx+wyz+xyz+w'x'z'$ 	15				
				8	13		
		$S_2(z, p)P_1[P_1(p)S_2(w, x)S_2(y, z)], p=P_2(w, x)$ $P_2(w', x')+P_2(z, q)+S_1[S_3(y, z, q)], q=S_2(w', x')$				10* 10	10

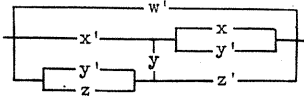
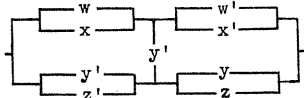
N	H	Switching Circuit	R	C	S	G	T
211 b	221	$wx'y + w'xy + wxz' + wx'z + w'xz$	20*				
			8	13			
		$S_2(w', x') S_2(z, p) P_1[P_3(y, z, p)], p = P_2(w', x')$				10	10
		$P_2(z, q) + S_1[S_1(q) + P_2(w, x) + P_2(y, z)], q = S_2(w, x)$					10
212 a	196	$wx + wyz' + w'yz + w'x'y'z'$	16				
			9	14			
		$P_2[p_2(w', x) S_2(y, z'), P_2[w, P_3(x, y, z)] S_2(y, z)]$				13*	
		$P_2[P_2(w', x) S_2(y, z'), P_1(w) S_2(y, z) S_3(x', y', z')]$					12
		$B_2[S_2(w, x') + P_2(y', z), S_1(w') + P_2(y', z') + P_3(x, y, z)]$					12
212 b	213	$wx'y + w'xy + w'xz' + wyz' + wy'z$	20*				
			9	14			
		$F_2[P_2(w, x') S_2(y', z), P_2[w', P_3(x', y', z')] S_2(y', z')]$				13	
		$F_2[P_2(w, x') S_2(y', z), P_1(w') S_2(y', z') S_3(x, y, z')]$					12
		$S_2[S_2(w', x) + P_2(y, z'), S_1(w) + P_2(y, z) + P_3(x', y', z')]$					12
213	209	$wxz' + w'xz + wyz + wy'z' + w'x'yz'$	21				
			10	16			
		$S_2(w, p) P_1[P_1(w) S_2(y, p) S_2(x, z)],$				14*	14
		$p = S_2(x, y) S_2(y, z) S_2(y', z')$					
		$P_2(w, q) + S_1[S_1(w) + P_2(x, q) + P_2(y, z')],$					14
		$q = P_2(x, y) + P_2(x, z') + P_2(x', z)$					
214 a	199	$wx + w'y'z' + x'yz$	11				
		$wx + (z + w'y')(z' + yz')$		8	12		
		$F_3[P_2(w', x'), P_3(w, y, z), P_3(x, y', z')]$				11	
		$P_1[S_2(w, x) S_3(w', y', z') S_3(x', y, z)]$					9
		$P_2(w', x') + P_3(w, y, z) + P_3(x, y', z')$					8
214 b	232	$wx'y' + w'xy + wyz' + xy'z$	16*				
		$[z(w' + y) + z'(x' + y')](w + x)$		8	12		
		$S_2(w', x') P_2[P_3(w', y', z'), P_3(x', y, z)]$				10	
		$S_2(w', x') S_3(w, y, z) S_3(x, y', z')$					8
		$S_1[P_2(w, x) + P_3(w', y', z') + P_3(x', y, z)]$					9

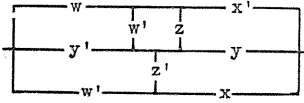
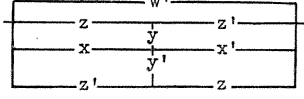
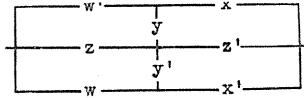
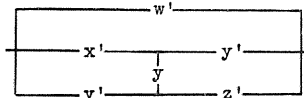
N	H	Switching Circuit	R	C	S	G	T
215 a	197	$wx + w'x'y' + wyz + xyz'$	15				
				8	13		
		$S_2[p, P_1(q)] S_2[q, P_1(p)],$				12	12
		$p = P_1(w) S_2(y, z'), q = P_1(x) S_2(y, z)$					
		$P_2(r, s) + S_1[S_2(r, s)],$					11
		$r = S_1(w') + P_2(y', z), s = S_1(x') + P_2(y', z')$					
215 b	230	$wx'y + w'xy + wy'z' + xy'z$	16*				
				8	13		
		$S_2(p, q) P_1[P_2(p, q)],$				11*	11
		$p = P_1(w') S_2(y', z), q = P_1(x') S_2(y', z')$					
		$P_2[r, S_1(s)] + P_2[s, S_1(r)],$					12
		$r = S_1(w) + P_2(y, z'), s = S_1(x) + P_2(y, z)$					
216	222	$wxy' + w'xy + wyz + x'y'z'$	16				
				9	15		
		$F_2[P_1(y') S_2(w', x') S_2(w, z'), P_1(y) S_2(w', x) S_2(x', z)]$				12*	12
		$B_2[S_1(y') + P_2(w', x') + S_2(x, z), S_1(y) + P_2(w, x')$					12
		$+ P_2(w', z')]$					
217 a	200	$wx + wyz + xy'z' + w'x'yz' + w'x'y'z$	21				
				10	15		
		$S_2[p, P_1(q)] S_2[q, P_1(p)],$				12*	12
		$p = P_1(w) S_2(y', z'), q = P_1(x) S_2(y, z)$					
		$P_2(r, s) + S_1[S_2(r, s)],$					11
		$r = S_1(w') + P_2(y, z), s = S_1(x') + P_2(y', z')$					
217 b	216	$wx'y' + w'xy + wx'z' + w'xz + wy'z' + xyz$	24*				
				10	15		
		$S_2(p, q) P_1[P_2(p, q)],$				11*	11
		$p = P_1(w') S_2(y, z), q = P_1(x') S_2(y', z')$					
		$P_2[r, S_1(s)] + P_2[s, S_1(r)],$					12
		$r = S_1(w) + P_2(y', z'), s = S_1(x) + P_2(y, z)$					

N	H	Switching Circuit	R	C	S	G	T
218 a	206	$wxy + wxz + wyz + w'y'z' + xyz + x'y'z'$	24				
				11	17		
		$S_2(p, q) P_1[P_2(p, q)],$ $p = S_2(w, x) P_1[P_2(y', z') S_2(w', x')], q = S_2(y', z')$ $P_2(r, s) + S_1[S_2(r, s)],$ $r = P_2(w', x') + S_1[S_2(y, z) + P_2(w, x)], s = S_2(y', z')$				14*	14
218 b	223	$wxy' + wyz' + wy'z + xyz' + xy'z + w'x'yz$	25*				
				11	17*		
		$S_2(p, q) P_1[P_2(p, q)],$ $p = S_2(w', x') P_1[P_2(y, z) S_2(w, x)], q = P_2(y', z')$ $P_2(r, s) + S_1[S_2(r, s)],$ $r = P_2(w, x) + S_1[S_2(y', z') + P_2(w', x')], s = P_2(y', z')$				14*	14
219	227	$wxy' + w'xy + wx'z + xyz' + w'x'y'z'$	21				
				10	16		
		$S_2(p, q) P_1[P_2(p, q)],$ $p = P_1(w') S_2(y, z'), q = S_2(x, y') S_2(x', y) S_2(x', z)$ $S_2(r, s) P_1[P_2(r, s)],$ $r = P_1(w) S_2(y, z'), s = S_2(x', z') S_3(x, y, z)$ $P_2(t, u) + S_1[S_2(t, u)],$ $t = S_1(w) + P_2(y', z), u = P_2(x', y') + P_3(x, y, z)$				14*	
220	212	$wxy + w'xy' + wxz + wyz + wx'y'z' + w'x'yz'$	26				
				11	17*		
		$S_2(p, q) P_1[P_2(p, q)],$ $p = P_1(w') S_2(y', z), q = S_2(x, y) S_2(x', y') S_2(y, z)$ $S_2(r, s) P_1[P_2(r, s)],$ $r = P_1(x) S_2(y, z), s = S_2(w', y) S_3(w, y', z')$ $P_2(t, u) + S_1[S_2(t, u)],$ $t = S_1(w) + P_2(y, z'), u = P_2(x', y) + P_3(x, y', z)$				14*	

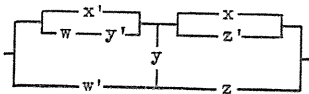
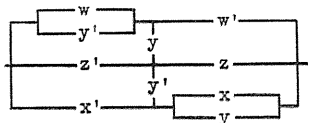
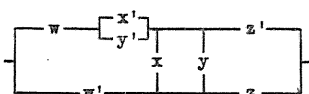
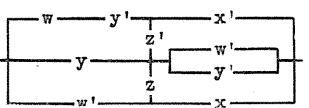
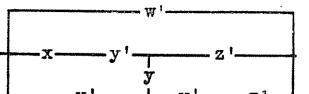


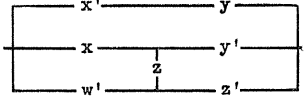
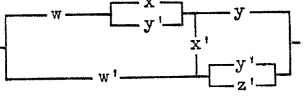
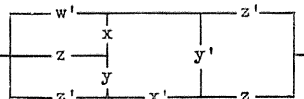
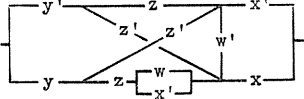
N	H	Switching Circuit	R	C	S	G	T
221	224	$wx'y' + w'xy + wxz + wx'z' + x'y'z'$	20				
				9	15		
		$S_2(p, q) P_1[P_2(p, q),$ $p = P_1(x) S(y', z), q = S_2(w', y') S_2(w, z')$ $P_2(r, s) + S_1[S_2(r, s)],$ $r = S_1(x') + P_2(y, z'), s = P_2(w, y') + P_2(w', z')$				12*	12
222 a	214	$w'x'y' + wxz + wyz + xyz' + x'y'z'$	20				
				10	15		
		$S_2(p, q) P_1[P_2(p, q)],$ $p = P_1(x) S_2(y, z), q = S_2(w', z) S_2(y', z')$ $P_2(r, s) + S_1[S_2(r, s)],$ $r = S_1(x') + P_2(y', z'), s = P_2(w', z') + P_2(y', z)$				12*	12
222 b	229	$wx'z' + wy'z' + xy'z + x'yz + w'xyz'$	21*				
				10	15		
		$S_2(p, q) P_1[P_2(p, q)],$ $p = P_1(x') S_2(y', z'), q = S_2(w', z') S_2(y', z)$ $P_2(r, s) + S_1[S_2(r, s)],$ $r = S_1(x) + P_2(y, z), s = P_2(w', z) + P_2(y', z')$				12*	12

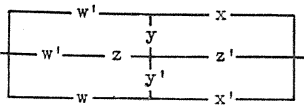
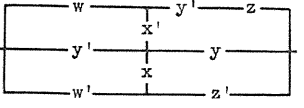
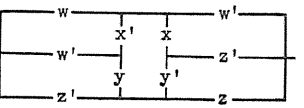
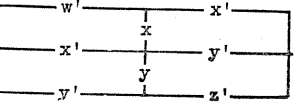
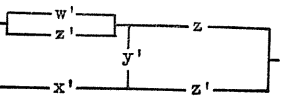
N	H	Switching Circuit	R	C	S	G	T
1'	401	1	0	0	0	0	0
2'	400	$w' + x' + y' + z'$ $F_4(w', x', y', z')$	4	4	8	4	4
3'	396	$w' + x' + y'$ $F_3(w', x', y')$	3	3	6	3	3
4'	397	$w' + x' + yz' + y'z$ $F_3[w', x', S_2(y, z) S_2(y', z')]$	8	6	10	7	7
5'	398	$w' + xy' + x'z + yz'$ $w' + (x + y + z)(x' + y' + z')$ $P_1[P_1(w') S_2(x, y') S_2(x', z) S_2(y, z')]$	12	7	11	8	8
6'	399	$wx' + w'z + xy' + yz'$ $(w + x + y + z)(w' + x' + y' + z')$ $P_1[S_2(w, x') S_2(w', z) S_2(x, y') S_2(y, z')]$ $S_4(w, x, y, z) S_4(w', x', y', z')$	12*	8	12	9	8
7'	390	$w' + x' + y'z'$ $F_3[w', x', P_2(y, z)]$	5	4	8	5	5
8'	391	$w' + xy' + x'y + x'z'$ $w' + (x' + y')(x + y + z')$ $F_2[w', S_2(x, y) P_1[P_3(x, y, z')]]$ $S_3(w, x, y) S_4(w, x', y', z)$	10*	6	10	8	7
9'	393	$w' + x'y' + x'z' + y'z' + xyz$	14				
				8	13		
		$F_2[w', S_2(x', p) P_1[P_2(x', p) S_2(y', z')]], p = P_2(y', z')$				11	11
10'	392	$w'x + wy' + w'z' + x'z$ $(w' + x' + y')(w + x + y + z')$ $P_2[P_3(w', x', y'), P_4(w, x, y, z')]$ $S_3(w, x, y) S_4(w', x', y', z)$	12*	7	11	9	7
11'	394	$wx' + w'x + w'y' + yz' + y'z$	15				
				9	14		
		$P_1[S_2(w, x') S_2(w', x) S_2(w', y') S_2(y, z') S_2(y', z)]$				11	11

N	H	Switching Circuit	R	C	S	G	T
12'	395	$wx' + w'y' + w'z' + y'z + xyz$	16*				
				9	14		
		$P_2[P_2(w', x') S_2(y, z) S_2(y', z'), P_1(w, x, y', z')]$				12	
		$P_1[P_2(w', x') S_2(y, z) S_2(y', z')] S_4(w', x', y, z)$					11
13'	371	$w' + x'$ $S_2(w, x)$	2	2	4		
						2	2
14'	378	$w' + xy' + x'y$ $F_2[w', S_2(x, y) S_2(x', y')]$	7	5	8		
						6	6
15'	384	$w' + xyz' + xy'z + x'yz + x'y'z'$	17				
				9	14		
		$F_2[w', S_2(x, p) S_2(x', P_1(p))], p = S_2(y, z') S_2(y', z)$				11	11
16'	383	$wx' + w'y + xy'$ $(w + x + y)(w' + x' + y')$ $P_1[S_2(w, x') S_2(w', y) S_2(x, y')]$ $S_3(w, x, y) S_3(w', x', y')$	9				
				6	9		
						7	
							6
17'	388	$wx' + w'x + yz' + y'z$ $P_1[S_2(w, x') S_2(w', x) S_2(y, z') S_2(y', z)]$	12	8	12		
						9	9
18'	389	$wx' + w'y'z' + w'y'z + xyz + xy'z'$	19				
				8	12		
		$S_2[P_2(w', x'), p] P_1[P_3(w, x, p)],$				12	
		$S_3(w, x, p) P_1[P_3(w, x, p)],$					11
		$p = S_2(y, z) S_2(y', z')$					
19'	372	$w' + x'y' + x'z' + y'z'$	10				
				6	11		
		$F_3[w', P_2(y, z), P_1(x) S_2(y, z)]$				8	8
20'	373	$w' + x'z + y'z'$ $F_2[w', S_2(x, z) S_2(y, z')]$	7	5	9		
						6	6

N	H	Switching Circuit	R	C	S	G	T
21'	374	$w' + x'y + x'z + xy'z'$	11				
				7	11		
		$F_2[w', S_2(x', p) S_2[x, P_1(p)]], p = P_2(y, z)$				9	9
22'	375	$wx' + w'x + w'y' + w'z' + y'z'$	15*				
				7	12		
		$P_2[P_2(w', x') S_2(y', z'), P_4(w, x, y', z')]$ $P_1[P_2(w', x') S_2(y', z')] S_4(w', x', y, z)$				10	9
23'	385	$w'x' + w'y' + w'z' + x'y' + x'z' + y'z' + wxyz$	23*				
				10	16		
		$S_2(w', p) P_1[P_2(w', p) S_2(y', z') S_2(x', q)],$ $p = P_2(x', q), q = S_2(y, z)$				13*	13
24'	376	$wx' + w'x + w'z + y'z'$ $wx' + w'(x + z) + y'z'$ $P_1[S_2(w, x') S_2(w', x) S_2(w', z) S_2(y', z')]$	12				
				7	11		
						9	9
25'	379	$w'x' + w'z + x'z' + y'z' + wxz'$	16*				
				8	13		
		$S_2(z, p) P_1[S_2(w', x') P_3(y, z, p)], p = P_2(w', x')$				10	10
26'	377	$w'x + x'y + x'z + wy'z'$	13				
				8	12		
		$P_2[P_2(w', x') S_2(y', z'), P_4(w, x, y, z)]$ $S_3(w, x, p) P_1[P_3(w, x, p)], p = S_2(y', z')$				10	9
27'	381	$w'y + x'y' + yz' + y'z$ $(w' + y' + z')(x' + y + z)$ $P_2[P_3(w', y', z'), P_3(x', y, z)]$ $S_3(w, y, z) S_3(x, y', z')$	12*				
				6	10	8	6

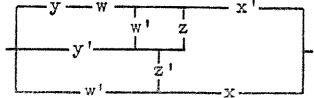
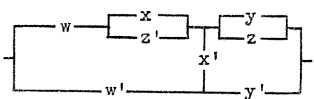
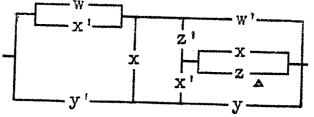
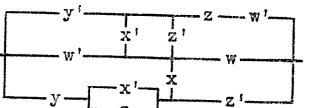
N	H	Switching Circuit	R	C	S	G	T
28'	380	$w'y + w'z + x'y + x'z' + wxy'$  $S_2(y, p)P_1[P_2(y, p)S_2(w', z)S_2(x', z')], p=P_2(w', x')$	16*		8	12	11* 11
29'	386	$w'y' + w'z' + x'y + x'z + wyz + xy'z'$  $S_2(y, p)P_1[P_3(w', y, p)S_2(x', z)], p=P_1(x')S_2(w, z)S_2(w', z')$	20		10	15	13* 13
30'	382	$w'y + w'z + xy' + x'y + wy'z'$  $P_2[P_3(w', x', y'), P_2(x, y)S_2(w, z')S_2(w', z)]$ $S_3(w, x, y)S_3[x', y', S_2(w, z')S_2(w', z)]$	16*		8	12	11 10
31'	387	$w'y + w'z + wx'y' + wx'z' + xyz + xy'z'$  $S_2(x', p)P_1[P_3(w', x', p)S_2(y', z')],$ $p=S_2(w, y')S_2(w', z)S_2(y, z')$	22*		10	16*	14 14
32'	344	$w' + x'y' + x'z'$ $w' + x'(y' + z')$ $F_2[w', P_1(x)S_2(y, z)]$	7		4	8	5 5
33'	351	$w' + x'y' + xyz'$ $w' + x'y' + xz'y$ $F_3[w', P_2(x, y), P_3(x', y', z)]$ $P_1[P_1(w')S_2(x', y')S_3(x, y, z')]$	8		6	10	8 7
34'	346	$w' + xy'z' + x'yz' + x'y'z$  $F_2[w', S_2(y, z)S_2(x', p)S_2[x, P_1(p)]], p=P_2(y, z)$	13		8	13	11 11

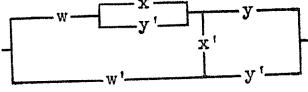
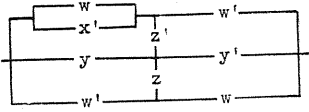
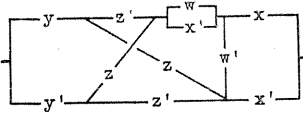
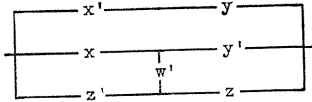
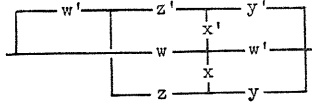
N	H	Switching Circuit	R	C	S	G	T
35'	345	$wx' + w'x + w'y' + w'z'$ $(w' + x')(w + x + y' + z')$ $S_2(w, x) P_1(P_4(w, x, y', z'))$ $S_2(w, x) S_4(w', x', y, z)$	12*		6 10	7	6
36'	359	$w'x' + xy' + x'y + w'z'$  $P_2[S_2(w', z') P_2(x', y'), P_3(w', x, y)]$ $P_1[S_2(w', z') P_2(x', y')] S_3(w, x', y')$	12		7 11	9	8
37'	363	$w'x' + w'y' + w'z' + x'y' + wxy$  $S_2(w, p) P_1[P_4(w, x', z', p)], p = S_2(x, y) S_2(x', y')$ $S_3(w, x, y') S_3(w, x', y) S_4(w', x, y, z)$	16*		8 13	11	10
38'	364	$w'y' + w'z' + xyz' + xy'z + x'yz + x'y'z'$  $S_2(z, p) P_1[P_3(w', z, p)], p = S_2(w', x') S_2(x, y') S_2(x', y)$ $S_3(w, z', q) P_1[P_2(z', q)], q = S_2(x, y) S_3(w, x', y')$	22*		9 16	12*	11
39'	368	$w'x' + w'y' + w'z' + xy'z' + x'yz' + x'y'z + wxyz$  $S_2(y', p) P_1[P_2(y', p) S_2(w', x') S_2(w', z')],$ $p = P_1(w') S_2(x, z') S_2(x', z)$	26*		11 17	14* 14	
40'	362	$wx' + w'x + w'y' + wyz'$ $(w + x + y')(w' + x' + yz')$ $P_2[P_2(w', x') S_2(y, z'), P_3(w, x, y')]$ $P_1[P_2(w', x') S_2(y, z')] S_3(w', x', y)$	13*		7 11	9	8
41'	361	$wx' + w'x + y'z' + w'yz$ $wx' + w'(x + yz) + y'z'$ $F_2[P_2(y, z), S_2(w, x) P_1[P_2(w, x) S_2(y, z)]]$ $P_1[S_2(w, x') S_2(w', x) S_2(y', z') S_3(w', y, z)]$	13		8 13	11	10

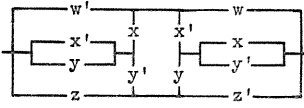
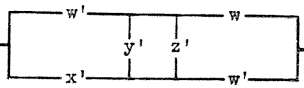
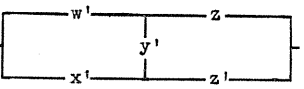
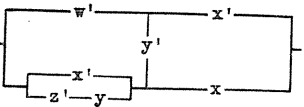
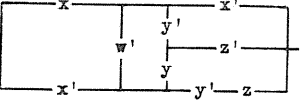
N	H	Switching Circuit	R	C	S	G	T
42'	360	$wx' + w'x + wy'z' + w'y'z + w'y'z$	18				
				9	14		
		$P_2[P_2(w', x') S_2(y', z'), P_2(w, x) S_2(y, z') S_2(y', z)]$				12	12
43'	365	$w'z' + wxy' + wx'y + w'xy + x'y'z$	19				
				9	14*		
		$S_2[P_2(w, z'), p] P_1[P_2(w', p) S_2(y', z)],$ $S_3(w', z, p) P_1[P_2(w', p) S_2(y', z)],$ $p = S_2(x, y) S_2(x', y')$				13*	12
44'	370	$w'x' + y'z' + wxy' + wxz' + w'yz + x'yz$	22				
				10	16		
		$S_2(p, q) P_1[P_2(p, q) S_2(w', x') S_2(y', z')],$ $p = P_2(w', x'), q = P_2(y', z')$				13*	13
45'	347	$w'x' + w'y' + w'z' + x'y' + x'z' + y'z'$	18*				
				8	14		
		$P_2[P_2(w', x') S_2(y', z'), S_2(w', x') P_2(y', z')]$				10	10
46'	348	$w'z + x'y' + x'z' + y'z'$	12				
				6	11		
		$P_2[P_2(w', z') S_2(x', y'), P_3(x', y', z)]$ $P_1[P_2(w', z') S_2(x', y')] S_3(x, y, z')$				9	8
47'	352	$w'y + x'z + y'z'$ $w'y + zx' + z'y'$ $P_1[S_2(w', y) S_2(x', z) S_2(y', z')]$	9	6	10	7	7

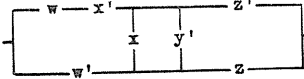
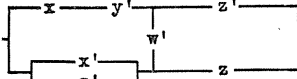
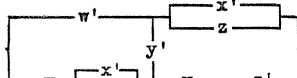
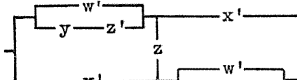
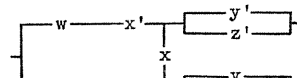
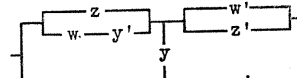
N	H	Switching Circuit	R	C	S	G	T
48'	349	$w'y + w'z + x'y' + x'z' + wy'z'$	16				
				9	14		
		$P_2[P_2(w', y') S_2(x', z'), P_2(x', y) S_2(w, z') S_2(w', z)]$				12	12
49'	353	$w'x' + w'y + w'z' + x'z' + wy'z$	16*				
				7	11		
		$S_2(w, p) P_1[P_3(w, x', p)], p = S_2(x', z') S_2(y', z)$				10*	10
50'	355	$w'x' + w'y' + w'z' + x'y' + x'z' + wxyz$	20*				
				9	14		
		$S_2(p, q) P_1[P_2(p, q)],$ $p = S_2(w, x), q = S_2(w', x') P_2(y', z')$				11	11
51'	350	$w'y + w'z + wx'y' + wx'z' + xy'z'$	18*				
				9	15		
		$P_2[P_1(w') S_2(x', y') S_2(x', z') S_2(y', z'), P_4(w, x, y, z)]$ $P_1[P_1(w') S_2(x', y') S_2(x', z') S_2(y', z')] S_4(w', x', y', z')$				13	12
52'	354	$w'x + w'y + x'z' + wy'z$ $x'z' + (w + x + y)(w' + y'z)$ $P_2[P_1(w') S_2(x', z') S_2(y', z), P_4(w, x, y, z')]$ $S_3(w, x, z') S_3(w, y, z) S_4(w', x', y', z)$	13		8	12	
						11	10
53'	358	$w'x + x'y' + x'z' + xyz$	13*				
				7	11		
		$S_2(x', p) P_1[P_3(w', x', p)], p = P_2(y', z')$				8*	8
54'	357	$w'x + w'z' + x'y' + wx'z + xyz'$	17				
				9	14		
		$S_2(x', p) P_1[P_3(w', x', p)], p = P_1(y') S_2(w, z) S_2(w', z')$				11*	11

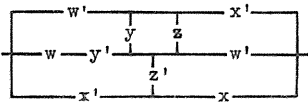
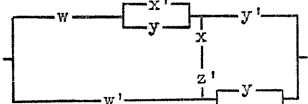
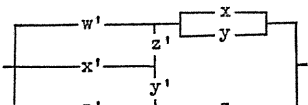
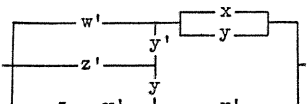
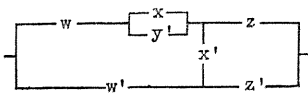
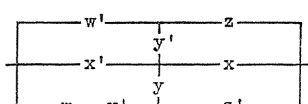


N	H	Switching Circuit	R	C	S	G	T
55'	356	$w'x + w'y' + wx'z + xy'z' + x'yz'$	18				
				9	14		
		$S_2(x', p) P_1[P_3(w', x', p)], p = S_2(w', y') S_2(w, z) S_2(y, z')$					12* 12
56'	367	$w'x' + w'y' + x'z' + wxy + wxz$	17*				
				8	12		
		$S_2(w', p) P_1[P_2(w', p) S_2(x, z) S_2(x', z')], p = P_2(x', y')$					11* 11
57'	366	$w'x' + w'y' + wxy + wxz' + wyz' + x'y'z$	22*				
				10	15		
		$S_2(x, p) P_1[P_3(w', x, p) S_2(y, z')], p = S_2(w, y) S_2(w', y') S_2(y', z')$					14* 14
59'	369	$wx'y' + w'xy' + wx'z' + w'x'z + wy'z' + w'yz' + wxyz$	29*				
				12	19		
		$S_2(p, q) P_1[P_2(p, q)], p = S_2(w', x') S_2(y, z), q = S_2(w, x') S_2(w', x) S_2(y', z')$					15* 15
59'	294	$w' + x'y'$ $S_2(w, x) S_2(w, y)$	4	3	6		4 4
60'	295	$w' + x'yz' + x'y'z$ $w' + x'(yz' + y'z)$ $F_2[w', P_1(x) S_2(y, z) S_2(y', z')]$	9		6 10		7 7
61'	315	$w' + xyz + x'y'z'$ $w' + (xy + x'z')(y' + z')$ $F_2[w', S_2(x, y') S_2(x', z) S_2(y, z')]$	9		7 11		8 8
62'	302	$wx' + w'x + w'y'$ $(w' + x')(w + x + y')$ $S_2(w, x) P_1[P_3(w, x, y')]$ $S_2(w, x) S_3(w', x', y)$	9*		5 8		6 5

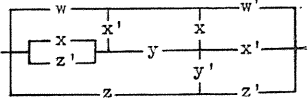
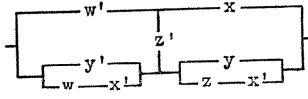
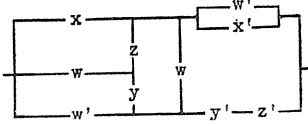
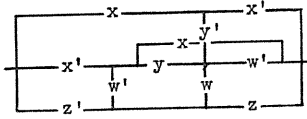
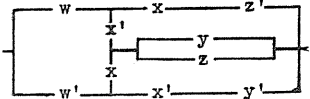
N	H	Switching Circuit	R	C	S	G	T
63'	337	$w'x' + w'y' + x'y' + wxy$	13*				
				7	11		
		$S_2[y, S_2(w', x') P_1(p)] S_2(y', p), p = P_2(w', x')$				9	9
64'	303	$wx' + w'x + w'yz + w'y'z'$ $(w' + x')(w + x + yz + y'z')$ $S_2(w, x) P_1[P_2(w, x) S_2(y, z) S_2(y', z')]$	14*				
				8	12	9*	9
65'	334	$w'x' + yz' + y'z$ $P_1[S_2(w', x') S_2(y, z') S_2(y', z)]$	9	6	10	7	7
66'	338	$w'x' + wyz + wy'z' + w'yz' + w'y'z$	19*				
				9	14		
		$S_2(w, p) P_1[P_3(w, x', p)], p = S_2(y, z) S_2(y', z')$				10	10
67'	342	$w'x' + w'yz + w'y'z' + x'yz + x'y'z' + wxyz' + wxy'z$	29				
				11	17		
		$S_2(p, q) P_1[P_2(p, q)],$ $p = S_2(w, x), q = S_2(w', x') S_2(y, z) S_2(x', z')$				13	13
68'	335	$xy' + x'y + w'yz + w'y'z'$	14				
				7	11		
		$P_2[P_2(x', y') S_2(w', z), P_2(x, y) S_2(w', z')]$				10	10
69'	339	$wxy + wx'y' + w'x'y + w'xz + w'y'z'$	20*				
				9	14*		
		$S_2(w, p) P_1[P_2(w, p) S_2(x, z) S_2(y', z')], p = S_2(x, y) S_2(x', y')$				13*	13
70'	336	$wx' + w'x + wyz + w'y'z'$ $(w + x + y'z')(w' + x' + zy)$ $P_2[P_2(w, x) S_2(y', z'), P_2(w', x') S_2(y, z)]$	14				
				8	13	10	10

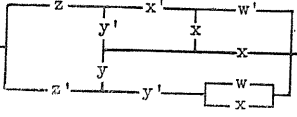
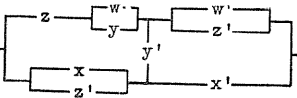
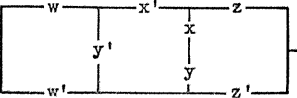
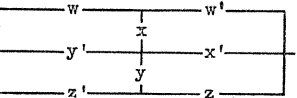
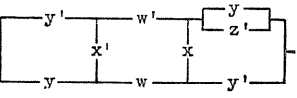
N	H	Switching Circuit	R	C	S	G	T
71'	343	$wxy + w'x'y' + wxz + w'x'z' + wyz + w'y'z' + xyz + x'y'z'$	32*				
			12	18			
		$S_2(p, q) P_1[P_2(p, q)],$ $p = S_2(w, x) S_2(y, z), q = S_2(w', x') S_2(y', z')$				13*	13
72'	296	$w'x' + w'y' + w'z' + x'y' + x'z'$	15*				
			6	11			
		$S_2(w, x) P_1[S_2(w', x') P_2(y', z')]$				7	7
73'	297	$w'y' + w'z + x'y' + x'z'$	12*				
			5	9			
		$S_2(w, x) P_1[P_1(y') S_2(w', z) S_2(x', z')]$				8	8
74'	298	$w'y + w'z + x'y' + x'z'$ $w'(y + z) + x'(y' + z')$ $F_2[P_1(w) S_2(y', z'), P_1(x) S_2(y, z)]$	12*				
				6	10		
						8	8
75'	299	$w'x + x'y' + x'z'$ $(w' + x')(x + y' + z')$ $S_2(w, x) P_1[P_3(x, y', z')]$ $S_2(w, x) S_3(x', y, z)$	9*				
				5	9		
						6	
							5
76'	309	$w'x' + w'y' + x'y' + xyz'$	13*				
			7	12			
		$S_2(z, p) P_1[P_2(w', p) S_2(x', y')], p = P_2(x', y')$				9*	9
77'	304	$w'x' + w'y' + w'z' + xy'z' + x'yz' + x'y'z$	21				
			9	15			
		$P_1[S_2(w', x') S_2(x, p) P_1[P_1(p) S_2(w, x) S_2(y, z)]], p = P_2(y, z)$				13*	13

N	H	Switching Circuit	R	C	S	G	T
78'	300	$w'x + w'z + x'y' + wx'z'$	13*				
			7	11			
		$S_2(w, x) P_1[P_2(x, y') S_2(w, z') S_2(w', z)]$				9*	9
79'	310	$w'y' + w'z' + x'z + xy'z'$	13				
			7	12			
		$P_2[P_2(w', z) S_2(x, y'), P_2(x', z') S_2(w', y')]$				10	10
80'	317	$w'x' + w'y' + w'z + x'y' + wxyz'$	17				
			9	14			
		$S_2(w', p) P_1[P_2(w', p) S_2(x', y')], p = P_3(x', y', z)$				10*	10
81'	305	$w'x' + w'z + xy'z' + x'yz' + x'y'z$	18*				
			9	15			
		$F_2[S_2(w, y') S_2(w, z) S_2(x, z'), P_1(y) S_2(x, z) S_2(x', z')]$ $S_3(x, y, z') S_2[w, S_2(z', p) S_2[z, P_1(p)]]], p = S_2(x', y')$				13	12
82'	301	$w'x + w'z + wx'y' + x'yz'$	14*				
			8	12			
		$F_2[P_1(w) S_2(x', z'), P_1(x) S_2(w', y') S_2(y, z)]$ $S_2(w, x) S_3(w, y, z) S_4(w', x', y', z')$				10*	9
83'	318	$w'x' + w'z + x'y' + xy'z'$ $x'y' + (w' + xz')(y + z)$ $S_2(y', p) P_1[P_2(w', p) S_2(x', y')], p = P_2(x', z)$	13*			7	11
						9	9
84'	316	$w'x' + w'y + w'z + wy'z' + x'yz$	17				
			8	13			
		$S_2(w, p) P_1[P_3(w, x', p)], p = S_2(y', z') P_1[P_3(x, y', z')]$ $S_2(w, q) P_1[P_3(w, x', q)], q = S_2(y', z') S_3(x', y, z)$				12*	11

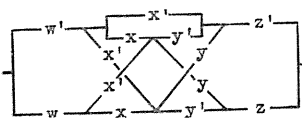
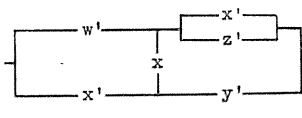
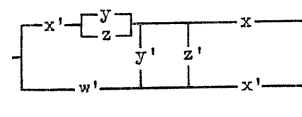
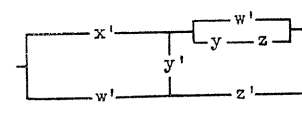
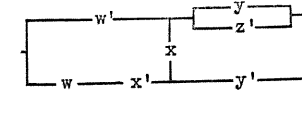
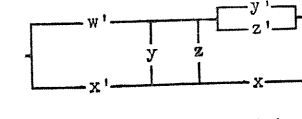
N	H	Switching Circuit	R	C	S	G	T
85'	306	$w'x' + w'y + w'z + x'yz' + x'y'z + wxy'z'$	22				
				10	16*		
		$S_2(p, q) P_1[P_2(p, q)],$ $p = S_2(w', x') P_2(y, z), q = F_2[w', P_1(x) S_2(y, z)]$				14*	14
86'	319	$w'x + w'y + w'z + wx'y' + xyz'$	17				
				9	14		
		$S_2[P_2(w, z), p] P_1[P_3[w', p, P_3(x', y', z)]]],$ $S_3(w', z', p) P_1[P_2(w', p) S_3(x, y, z')],$ $p = P_2(x, y)$				13	11
87'	307	$w'x + w'y + xy'z' + x'yz' + x'y'z$	18*				
				8	15		
				9	14*		
		$S_2(x', p) P_1[P_2(w', p) S_2(x', y') S_2(x', z')], p = P_2(y, z)$				11*	11
88'	321	$w'z' + x'z + wxy'$ $wy'x + (w' + z)(x' + z')$ $F_2[S_2(w, z') S_2(x, z), P_3(w', x', y)]$ $P_1[S_2(w', z') S_2(x', z) S_3(w, x, y')]$	10	7	11*	9	8
89'	325	$w'x' + w'z' + x'y' + wxz$	13*				
				7	11		
		$S_2(w', p) P_1[P_2(w', p) S_2(x', y')], p = P_2(x', z')$				9*	9
90'	312	$w'z + wx'y' + xy'z' + x'yz'$	15				
				9	14*		
		$P_1[S_2(w', z) S_2(w, p) P_1[P_2(z, p) S_2(x, y)]]], p = P_2(x, y)$				12	12

N	H	Switching Circuit	R	C	S	G	T
91'	313	$w'z + x'y' + w'xy + wx'z' + wy'z'$	18				
				9	14		
		$P_1[S_2(x', y') S_2(w, p) S_2[w', P_1(p)]], p = P_1(z) S_2(x, y)$				11*	11
92'	323	$x'y' + wx'z + w'xz + wy'z' + w'yz'$	19*				
				9	14		
		$S_2(w, p) P_1[P_2(w, p) S_2(x', y')], p = S_2(x', z) S_2(y', z')$				11*	11
93'	328	$w'y' + x'y' + wx'z' + w'xz' + w'x'z + wxyz$	23				
				10	15		
		$S_2(w', p) P_1[P_2(w', p) S_2(x', y')], p = P_1(y') S_2(x, z') S_2(x', z)$				12*	12
94'	314	$w'y + w'z + wx'y' + wx'z' + wy'z'$	18*				
				9	14		
		$S_2(w', p) P_1[P_2(w', p) S_2(x', y') S_2(x', z')], p = P_2(y, z)$				11	11
95'	333	$w'y' + x'y + wyz' + xy'z$ $(w' + y + zx)(x' + y' + wz')$ $S_2(y, p) P_1[P_3(w', y, p)], p = P_1(x') S_2(w, z')$	14				
				8	13		
						9*	9
96'	322	$w'y + x'z + wy'z' + xy'z'$ $yw' + zx' + y'z'(w + x)$ $P_1[S_2(w', y) S_2(x', z) P_1[S_2(w', x') P_2(y, z)]]$	14				
				8	13		
						10	10
97'	330	$w'x' + w'y' + wxy + wyz' + x'y'z$	18*				
				9	14		
		$S_2(y', p) P_1[P_2(y', p) S_2(w', x')], p = P_1(w') S_2(x', z)$				10*	10
98'	324	$w'y + wx'z + w'xz + w'x'z' + wy'z'$	19*				
				9	14		
		$S_2(w, p) P_1[P_3(w, y, p) S_2(x', z')], p = S_2(x', z) S_2(y', z')$				12*	12

N	H	Switching Circuit	R	C	S	G	T
99'	327	$wx'y + w'x'y' + wx'z + w'xz + w'yz' + wxy'z'$	25*				
			11	17			
		$S_2(w', p) P_1[P_2(w', p) S_2(x', y) S_2(x', z)],$ $p = S_2(x, y) S_2(x', z') S_2(y', z)$			15*	15	
100'	326	$w'x + w'y + wx'z + wy'z'$ $(w + x + y)(w' + x'z + y'z')$ $P_2[P_1(w') S_2(x', z) S_2(y', z'), P_3(w, x, y)]$ $S_3(w', x', y') S_3(w, x, z) S_3(w, y, z')$	14*	8	12	10	9
101'	331	$w'x + wx'y + xy'z' + x'yz' + x'y'z$	19*				
			9	14			
		$S_2(x', p) P_1[P_3(w', x', p)], p = S_2(w, y) S_2(y, z') S_2(y', z)$ $S_3(w, x, q) P_1[P_2(x, q)], q = S_2(y', z') S_3(w', y, z)$			12*		11
102'	332	$w'x + wx'y + wx'z + w'yz + w'y'z' + xy'z'$	23				
			10	16			
		$S_2(x, p) P_1[P_2(x, p) S_2(y, z) S_2(w', q)],$ $p = P_2(w', q), q = P_2(y, z)$			13*	13	
103'	340	$w'xy' + w'x'z + w'yz' + xyz' + xy'z + x'yz + wx'y'z'$	29*				
			11	17			
		$S_2(y, p) P_1[P_2(y, p) S_2(w', x) S_2(w', z)],$ $p = S_2(w', x') S_2(x, z') S_2(x', z)$			15*	15	
104'	341	$wx'y + w'xy + w'x'y' + wxz' + wx'z + w'xz$	24*				
			10	15			
		$S_2(y, p) S_2(z, q) P_1[P_4(y, z, p, q)],$ $p = P_2(w, x), q = P_2(w', x')$			13*	13	

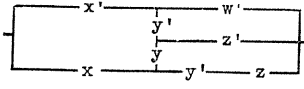
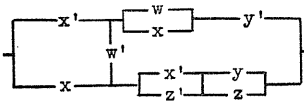
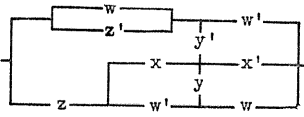
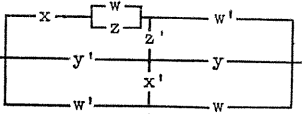
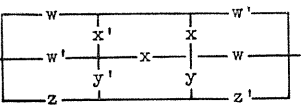
N	H	Switching Circuit	R	C	S	G	T
105'	308	$wx'y' + w'xy' + wx'z' + w'x'z + wy'z' + w'yz'$	24*				
			11	17*			
		$F_2[S_2(w, y) S_2(x, z) S_2(y', z'), S_2(w', x') S_2(x, y) S_2(w, z)]$				14	
		$S_3(w, x, p) S_3(y, z, q) P_1[P_2(p, q)], p = F_2(y, z), q = F_2(w, x)$					13
106'	311	$x'z' + y'z' + wx'y' + w'xz + w'yz$	18*				
			9	14			
		$S_2(p, q) P_1[P_2(p, q)],$				13	
		$p = S_2(w, z), q = S_2(x', y) S_2(x, z) S_2(y', z')$					
		$S_3(x, y, z') P_1[P_1(z') S_2(w, r) S_2[w', P_1(r)]]], r = P_2(x, y)$					12
107'	320	$w'z' + x'y' + y'z' + w'xy + wx'z$	17				
			8	12			
		$S_2(p, q) P_1[P_2(p, q)], p = S_2(w', y), q = S_2(x', z) S_2(y', z')$				11*	11
108'	329	$w'y' + w'z' + x'y' + x'z' + wxyz$	17*				
			8	12			
		$S_2(p, q) P_1[P_2(p, q)], p = S_2(w, x), q = P_2(y', z')$				9	9
109'	238	$w' + x'y'z'$	5	4	8		
		$F_2[w', P_3(x, y, z)]$				5	5
110'	257	$wx' + w'x + w'y'z'$	10				
		$(w' + x')(w + x + y'z')$		6	10		
		$S_2(w, x) P_1[P_2(w, x) S_2(y', z')]$				7	7
111'	292	$wxy + wx'y' + w'xy' + w'x'y + w'x'z'$	20*				
			9	16			
		$S_2(w, p) P_1[P_2(w, p) S_2(y', z')], p = S_2(x, y) S_2(x', y')$				11	11

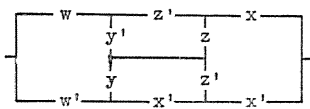
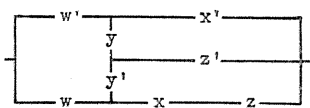
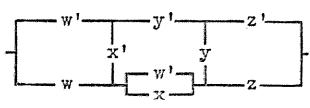
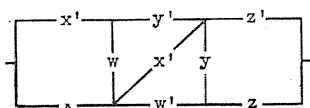
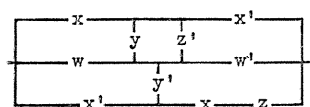
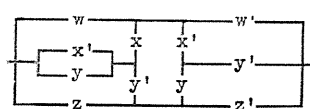


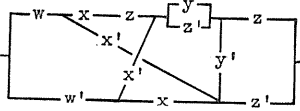
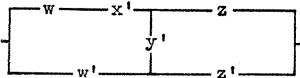
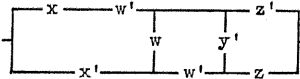
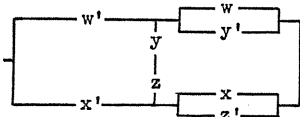
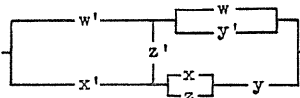
N	H	Switching Circuit	R	C	S	G	T
112'	293	$w'x'y' + w'x'z' + w'y'z' + x'y'z' + wxyz' + wxy'z$ $+ wx'yz + w'xyz$ 	36*				
				13	20		
		$S_2(p, q)P_1[P_2(p, q)S_2(w', y')]$ , $p = S_2(w, x)S_2(w', x'), q = S_2(y, z)S_2(y', z')$				15	15
113'	239	$w'x' + w'y' + w'z' + x'y'$ 	12*				
				6	11		
		$S_2(w, x)S_2(w, y)P_1[P_3(x', y', z')]$ $S_2(w, x)S_2(w, y)S_3(x, y, z)$				8	7
114'	240	$w'y + w'z' + x'y'$ $w'(y + z') + x'y'$ $F_2[P_2(x, y), P_1(w)S_2(y', z)]$	9		5	9	
						7	7
115'	242	$w'x' + w'y' + w'z' + x'yz' + x'y'z$ 	17*				
				8	13		
		$P_2[P_1(w')S_2(x', y)S_2(x', z), S_2(w', x')P_2(y', z')]$ $S_2(w, x)P_1[S_2(w', x')P_2(y', z')]S_3(w, y', z')$				11	10
116'	243	$w'y' + w'z' + x'yz' + x'y'z'$ 	14				
				7	12		
		$F_2[P_1(w)S_2(y, z), P_1(x)S_2(y, z')S_2(y', z)]$				10*	10
117'	241	$w'x + w'y + w'z' + w'z' + wx'y'$ 	13*				
				7	11		
		$S_2(w, p)P_1[P_3(w, z', p)], p = S_2(x', y')$				8	8
118'	244	$w'x + w'y' + x'yz' + x'y'z'$ 	14*				
				7	11		
		$S_2(z, p)P_1[P_2(w', p)S_2(x', z)], p = P_2(x, y')$ $S_2(w, x)S_3(w, y', z')S_3(x', y, z)$				9*	8

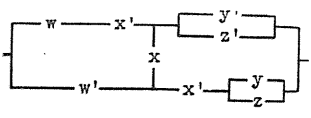
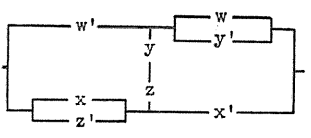
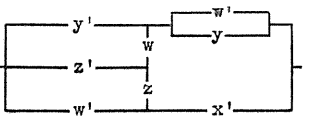
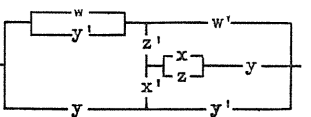
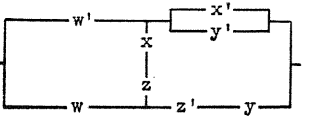
N	H	Switching Circuit	R	C	S	G	T
119'	266	$w'y' + w'z' + xy'z' + x'yz$	14				
				8	13*		
		$P_2[P_3(w', y', z), S_2(w', y') S_2(x', y) S_2(x, z')]$ $S_2(w, y, z') P_1[S_2(w', y') S_2(x', y) S_2(x, z')]$				11	10
120'	245	$w'x + w'y' + w'z' + x'y'z' + wx'yz$	18*				
				9	14		
		$S_2[P_2(w, y'), p] P_1[P_2(w', p)], p = P_1(x) S_2(y, z') S_2(y', z)$ $S_3(w', y, p) P_1[P_2(w', p)], p = P_1(x) S_2(y, z') S_2(y', z)$				12	11
121'	276	$w'x' + w'y' + w'z' + x'y'z' + wxyz$	18*				
				9	14		
		$S_2(w', p) P_1[P_3[w', p, P_3(x, y, z)], p = P_3(x', y', z')$ $S_2(w', p) P_1[P_2(w', p) S_3(x', y', z')], p = P_3(x', y', z')$				12*	11
122'	267	$w'x' + w'y' + x'y' + wxyz'$	14				
				8	13		
		$S_2(w', p) S_2[w, S_2(x', y') S_2(z', p)], p = P_2(x', y')$				10	10
123'	253	$w'y + x'y' + wy'z'$ $(w' + y') (x' + y + wz')$ $S_2(w, y) P_1[P_2(x', y) S_2(w, z')]$	10				
				6	10		
						7	7
124'	270	$w'x' + y'z' + w'yz + x'yz$	14				
				8	13		
		$P_1[S_2(w', x') S_2(y', z') P_1[S_2(w, x) P_2(y', z')]]$				10	10
125'	260	$w'x' + wy'z' + w'yz' + w'y'z + x'yz$	19*				
				8	13		
		$S_2(w, p) P_1[P_2(x', p) S_2(w, y')], p = S_2(y, z) S_2(y', z')$				11*	11

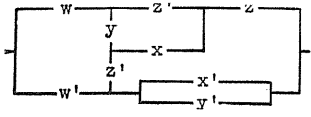
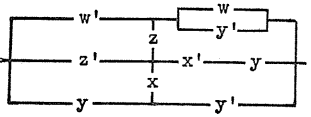
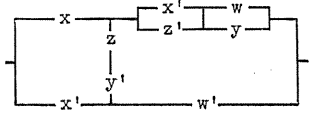
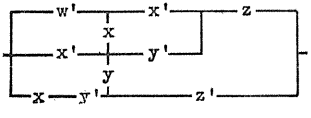
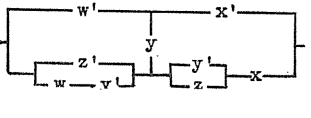
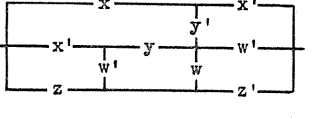
N	H	Switching Circuit	R	C	S	G	T
126'	258	$w'x' + w'y'z' + w'y'z + x'y'z' + x'y'z + wxy'z'$	24				
				10	16		
		$S_2(p, q) P_1[P_1(p) S_2(w', x') S_2(q, S_2(y, z))],$ $p = P_2(w', x'), q = S_2(y', z')$				14*	14
127'	256	$w'x + w'y' + wx'y + wx'z'$	14*				
				7	11		
		$S_2(w', p) P_1[P_2(w', p) S_2(x', z')], p = P_2(x', y')$ $S_2(w, x) S_3(w', x', y) S_3(w, y', z)$				9	8
128'	254	$w'x + wx'y' + x'y'z' + x'y'z$ $(w' + x')[x + (w + y + z)(y' + z')]$ $S_2(w, x) S_2[x', S_2(w, y') S_2(y, z') S_2(y', z)]$ $S_2(w, x) S_3(x', y, z) S_4(w', x', y', z')$	15*	8	12	10	9
129'	282	$w'z' + x'y' + xyz$ $w'z' + (x + y')(x' + yz)$ $P_1[S_2(w', z') S_2(x', y') P_1[P_2(x', y', z')]]$ $P_1[S_2(w', z') S_2(x', y') S_3(x, y, z)]$	10	7	11	9	8
130'	272	$x'y' + w'xz' + w'yz + xyz'$	15				
				8	13		
		$P_2[P_2(x', y) S_2(w', z'), P_1(y') S_2(w', z) S_2(x, z')]$				11	11
131'	279	$w'x + wx'y' + wx'z' + w'yz + w'y'z'$	19*				
				9	14		
		$S_2(w, p) P_1[P_2(w, p) S_2(y', z')], p = S_2(x', y') S_2(x', z')$				11*	11
132'	279	$w'x' + w'y' + wxy + x'y'z'$	14				
				8	13		
		$S_2(w', p) S_2[w, P_2[p, P_3(x, y, z)]], p = P_2(x', y')$ $S_2(w', p) P_1[P_2(w', p) S_3(x', y', z')], p = P_2(x', y')$				11	10

N	H	Switching Circuit	R	C	S	G	T
133'	263	$w'x' + xyz' + xy'z + xy'z + x'y'z'$	15				
				8	13		
		$P_2[P_1(x') S_2(y, z') S_2(y', z), P_2(w', x) S_2(y', z')]$				11	11
134'	269	$wx'y' + w'xy' + w'x'z + w'yz' + xyz'$	20*				
				10	15		
		$F_2[S_2(w, x') S_2(x, z) S_2(y', z'), P_1(y) S_2(w, x) S_2(w', x')]$				13*	13
135'	281	$w'z' + wx'y' + w'xy' + w'x'y + wxyz$	20				
				10	15*		
		$S_2(w, p) P_1[P_3(w, z', p)], p = S_2(x', y') P_1[P_3(x', y', z')]$ $S_2(w, q) P_1[P_3(w, z', q)], q = S_2(x', y') S_3(x, y, z)$				12	11
136'	277	$wx'y' + w'xy' + w'x'z' + w'yz' + wxyz'$	21				
				10	16		
		$S_2(w, p) P_1[P_2(w, p) S_2(y', z')],$ $p = S_2(x', y') P_1[P_3(x', y', z')]$ $S_2(w, q) P_1[P_2(w, q) S_2(y', z')],$ $q = S_2(x', y') S_3(x, y, z)$				13*	12
137'	264	$w'x' + w'yz + w'y'z' + x'y'z' + wxyz' + wxy'z$	25				
				11	17*		
		$S_2(w', p) P_1[P_2(w', p) S_2(x', q)],$ $p = P_2(x', q) S_2(y, z), q = P_2(y, z)$				13*	13
138'	285	$w'x' + wxy' + wy'z + xyz'$ $w'x' + (w+y)(y'+z')(x+z)$ $P_1[S_2(w', x') P_1[S_2(w', y') S_2(x', z') S_2(y, z)]]$	15				
				8	12		
						10*	10

N	H	Switching Circuit	R	C	S	G	T
139'	275	$w'x' + wxy' + wxz' + wy'z' + w'yz$	19				
				10	16		
		$S_2(w', p) P_1[P_2(w', p) S_2(y', z')], p = P_1(x') S_2(y, z)$				10*	10
140'	287	$w'x' + wxz + wy'z' + w'yz'$	15				
				8	13		
		$S_2(w, p) P_1[P_3(w, x', p)], p = S_2(x, z) S_2(y', z')$				10	10
141'	291	$w'x' + wxy + wxz + w'y'z' + x'y'z'$	19				
				9	14		
		$S_2(p, q) P_1[P_2(p, q)], p = S_2(w', x') S_2(y', z'), q = S_2(w, x)$				11	11
142'	286	$wx'y + w'xy + w'x'y' + w'xz + wy'z'$	20*				
				9	14*		
		$S_2(w, p) P_1[P_2(w, p) S_2(x', y')], p = S_2(x', y) S_2(y', z')$				11	11
143'	288	$wx'y + w'xy + w'x'y' + wx'z' + w'xz' + wxy'z$	25*				
				10	16		
		$S_2(w', p) P_1[P_2(w', p) S_2(x', z')], p = S_2(x, y) S_2(x', y') S_2(x, z')$				13	13
144'	274	$wx'y' + w'xy + wx'z' + w'xz + wy'z' + w'yz + x'y'z'$	28*				
				11	17		
		$S_2(w, p) P_1[P_2(w, p) S_2(x', q)], p = P_1(q) S_2(x', y') S_2(x', z'), q = P_2(y, z)$				14*	14

N	H	Switching Circuit	R	C	S	G	T
145'	290	$wx'y'+w'xy'+wx'z'+w'xz'+w'y'z'+wxyz+w'x'yz$  $S_2(p, q) P_1[P_2(p, q)],$ $p=P_2(y', z'), q=S_2(w, x) P_1[P_2(w, x) S_2(y', z')]$	30*				
				12	18*		
						14	14
146'	246	$w'y'+w'z'+x'y'+x'z'$ $(w'+x')(y'+z')$ $S_2(w, x) S_2(y, z)$	12*				
				4	8		
						4	4
147'	249	$w'y'+w'z'+x'y'+wx'z$  $S_2(w, x) S_2[y, S_2(w, z) S_2(w', z')]$	13*				
				6	10		
						8	8
148'	247	$w'y'+x'y'+wx'z'+w'xz'+w'x'z$  $S_2(w, x) P_1[P_1(y') S_2(z, p) S_2[z', P_1(p)]], p=P_2(w, x)$	18*				
				8	13		
						11*	11
149'	251	$w'y'+w'z'+wx'y+wx'z$ $wx'(y+z)+w'(y'+z')$ $F_2[P_1(w) S_2(y, z), P_2(w', x) S_2(y', z')]$ $S_2(w, x) S_3(w, y', z') S_3(w', y, z)$	14*				
				7	11		
						9	
							8
150'	252	$w'y'+x'z'+wx'y+w'xz$  $S_2(w, x) P_1[S_2(w, y) S_2(w', y') S_2(x, z) S_2(x', z')]$	14				
				8	12*		
						11*	11
151'	250	$w'y'+wx'z'+w'xz'+x'yz$  $F_2[P_1(w) S_2(x', y) S_2(y, z), P_1(x) S_2(w', z') S_2(y', z)]$	15				
				8	12		
						12	12

N	H	Switching Circuit	R	C	S	G	T
152'	248	$wx'y' + w'xy' + wx'z' + w'x'z + w'yz'$	20*				
				9	14		
		$F_2[S_2(w, x) S_2(w', x') S_2(y, z), P_2(w, x) S_2(y', z')]$ $S_2(w, x) P_1[S_2(w', x') P_2(y', z')] S_4(w', x', y', z')$				12	11
153'	280	$w'x' + w'y' + x'z' + wxyz$	14				
				8	12		
		$S_2[S_2(w, z), p] P_1[P_2(w', p) S_2(x', z')], p = P_2(x', y')$				11	11
154'	268	$w'x' + w'y' + wyz' + x'y'z$	14				
				8			
		$P_2[P_1(y') S_2(w', x') S_2(w, z'), P_2(w', y) S_2(x', z)]$				11*	11
155'	261	$w'x' + w'y' + x'yz + x'y'z' + wxyz'$	19				
				10	15		
		$S_2(p, q) P_1[P_2(p, q)],$ $p = S_2(w, y') S_2(x, y), q = P_2(w', z) S_2(x, y')$				13*	13
156'	283	$w'x' + w'y' + wyz' + xy'z$	14				
				8	12		
		$P_2[P_3(w', y, z), S_2(w', x') S_2(w, z') S_2(x, y')]$ $S_3(w, y', z') P_1[S_2(w', x') S_2(w, z') S_2(x, y')]$				11	10

N	H	Switching Circuit	R	C	S	G	T
157'	265	$w'x' + w'y' + xy'z' + x'yz' + wxyz$	19				
				9	16*		
				10	15*		
		$S_2(p, q) P_1[P_2(p, q)],$ $p = S_2(x, y), q = P_1(w') S_2(x, z') S_2(y, z')$				12*	12
158'	271	$w'x' + wxz' + w'yz' + w'y'z + x'y'z$	19				
				9	15		
		$P_1[S_2(w', x') S_2(z, p) P_1[P_3(x', z, p)]], p = P_1(y) S_2(w, x)$				12*	12
159'	259	$w'xy' + w'x'z + w'yz' + xy'z' + x'y'z' + x'y'z$	24*				
				10	16		
		$S_2(x', p) P_2[P_3(w', x', p), S_2(w', x') P_2(y', z')], p = P_2(y, z)$				13	13
160'	278	$w'x' + wxy' + w'y'z + xy'z' + x'yz'$	19*				
				9	14*		
		$S_2(p, q) P_1[P_2(p, q)],$ $p = P_2(x', y'), q = S_2(w, y') S_2(w, z) S_2(x, z')$				13*	13
161'	262	$w'xz + w'yz + w'y'z' + xy'z' + x'yz' + wx'y'z$	25*				
				10	16		
		$S_2(p, q) P_1[P_2(p, q)],$ $p = P_2(x, y) S_2(w', z'), q = S_2(w, z) P_1[P_3(x', y', z)]$ $S_2(r, s) P_1[P_2(r, s)],$ $r = P_2(x, y) S_2(w', z'), s = S_2(w, z) S_3(x, y, z')$				15*	14



N	H	Switching Circuit	R	C	S	G	T
162'	273	$wx'y' + w'x'y + w'xz + xy'z' + x'yz'$	20*				
				9	14		
		$S_2(w', p) P_2[P_3(w', z', p), P_3(x', y', z)],$ $S_2(w', p) P_1[P_3(w', z', p)] S_3(x, y, z'),$ $p = P_2(x, y)$				12	11
163'	284	$w'xy + wx'z + wy'z' + w'yz' + xy'z' + x'y'z$	24*				
				10	16		
		$S_2(p, q) P_1[P_2(p, q)],$ $p = P_1(w) S_2(x, y') S_2(y', z'), q = S_2(x, z) S_2(y, z')$				14*	14
164'	289	$wx'y' + w'xy' + w'yz' + w'y'z + x'yz' + x'y'z + wxyz$	29*				
				10	16		
		$S_2(p, q) P_2[P_2(p, q), P_4(w, x, y, z)],$ $S_2(p, q) P_1[P_2(p, q)] S_4(w', x', y', z'),$ $p = P_2(w', x'), q = S_2(y, z)$				14*	13

## References

- 1) Birkhoff, G.: Lattice Theory, American Mathematical Society, New York, 1940.
- 2) Birkhoff, G. and Mac Lane, S.: A Survey of Modern Algebra, Mac Millan, New York, 1953.
- 3) Bowden, B. V.: Faster Than Thought, Pitman, 1953.
- 4) Caldwell, S. H.: Switching Circuits and Logical Design, Wiley, New York, 1958.
- 5) Caldwell, S. H.: The Recognition and Identification of Symmetric Switching Functions, Trans. AIEE, Vol. **73**, Pt. 1, pp. 142-147, 1954.
- 6) Davis, R. L.: The Number of Structures of Finite Relations, Proc. Amer. Math. Soc., Vol. **4**, pp. 486-495, 1953.
- 7) Durst, L. K.: On Certain Subsets of Finite Boolean Algebras, Proc. Amer. Math. Soc., Vol. **6**, pp. 695-697, 1955.
- 8) Gilbert, E. N.: Lattice Theoretic Properties of Frontal Switching Functions, Jour. Math. and Phys., Vol. **33**, pp. 52-67, 1954.
- 9) Higonnet, R. A. and Grea, R. A.: Logical Design of Electrical Circuits, McGraw-Hill, New York, 1958.
- 10) Hohn, F. E.: Some Mathematical Aspects of Switching, Amer. Math. Month., Vol. **62**, pp. 75-90, 1955.
- 11) Hohn, F. E. and Schissler, L. R.: Boolean Matrices and the Design of Combinational Relay Switching Circuits, Bell System Tech. Jour., Vol. **34**, pp. 177-202, 1955.
- 12) Hunter, L. P.: Handbook of Semiconductor Electronics, McGraw-Hill, New York, 1956.
- 13) Karnaugh, M.: The Map Method for Synthesis of Combinational Logic Circuits, Trans. AIEE, Vol. **72**, Pt. 1, pp. 593-599, 1953.
- 14) Keister, W., Ritchie, A. E. and Washburn, S. H.: The Design of Switching Circuits, Van Nostrand, New York, 1951.
- 15) Lee, C. Y.: Switching Functions on an  $N$ -Dimensional Cube, Trans. AIEE, Vol. **73**, Pt. 1, pp. 289-291, 1954.
- 16) Luce, R. D.: A Note on Boolean Matrix Theory, Proc. Amer. Math. Soc., Vol. **3**, pp. 382-388, 1952.
- 17) Lunts, A. G.: The Application of Boolean Matrix Algebra to the Analysis and Synthesis of Relay Contact Networks, Dokl. Akad. Nauk SSSR, T. 70, pp. 421-423, 1950.
- 18) Montgomerie, G. A.: Sketch for an Algebra of Relay and Contactor Circuits, Jour. IEE, Vol. **95**, Pt. II, pp. 355-365, 1948.
- 19) Muller, D. E.: Application of Boolean Algebra to Switching Circuit Design and Error Detection, Trans. IRE, EC-3, pp. 6-12, 1954.
- 20) Muller, D. E.: Boolean Algebras in Electric Circuit Design, Amer. Math. Month., Vol. **61**, No. 7, Pt. 2, pp. 27-28, 1954.
- 21) McCluskey, E. J. Jr.: Minimization of Boolean Functions, Bell System Tech. Jour., Vol. **35**, pp. 1417-1444, 1956.
- 22) Ninomiya, I.: On the Number of Types of Symmetric Boolean Output Matrices, Memoirs Fac. Eng. Nagoya Univ., Vol. **7**, No. 2, pp. 115-124, 1955.
- 23) Ninomiya, I.: A Theory of the Coordinate Representation of Switching Functions, Memoirs Fac. Eng. Nagoya Univ., Vol. **10**, No. 2, pp. 176-190, 1958.
- 24) Ninomiya, I.: On the Number of Genera of Boolean Functions of  $n$  Variables, Memoirs Fac. Eng. Nagoya Univ., Vol. **11**, No. 1-2, pp. 54-58, 1959.
- 25) Ninomiya, I.: On the number of Types of Boolean Functions of Moderate Dimensions, Unpublished.
- 26) Petrick, S. R.: A Direct Determination of the Irredundant Forms of Boolean Function from the Set of Prime Implicants, AFCRC-TR-56-110, 1956.
- 27) Phister, M.: Logical Design of Digital Computers, Wiley, New York, 1958.
- 28) Pólya, G.: Sur les types des propositions composées, Jour. Symbolic Logic, Vol. **5**, pp. 98-103, 1940.

- 29) Povarov, G. N.: On the Functional Separability of Boolean Functions, Dokl. Akad. Nauk SSSR, T. 94, pp. 801-804, 1954.
- 30) Quine, W. V.: The Problem of Simplifying Truth Functions, Amer. Math. Month., Vol. 59, pp. 521-531, 1952.
- 31) Quine, W. V.: A Way to Simplify Truth Functions, Amer. Math. Month., Vol. 62, pp. 627-631, 1955.
- 32) Quine, W. V.: On Cores and Prime Implicants of Truth Functions, Amer. Math. Month., Vol. 66, pp. 755-760, 1959.
- 33) Richards, R. K.: Arithmetic Operations in Digital Computers, Van Nostrand, New York, 1955.
- 34) Somson, E. W. and Mills, B. E.: Circuit Minimization; Algebra and Algorithms for New Boolean Canonical Expressions, AFCRC-TR-54-21, 1951.
- 35) Shannon, C. E.: A Symbolic Analysis of Relay and Switching Circuits, Trans. AIEE. Vol. 57, pp. 713-723, 1938.
- 36) Shannon, C. E.: The Synthesis of Two-Terminal Switching Circuits, Bell System Tech. Jour., Vol. 28, pp. 59-98, 1949.
- 37) Slepian, E.: On the Number of Symmetry Types of Boolean Functions of  $n$  Variables, Canadian Jour. Math., Vol. 5, pp. 158-193, 1954.
- 38) Staff of the Computation Laboratory of Harvard University: Synthesis of Electronic Computing and Control Circuits, Harvard University Press, Cambridge, Mass., 1951.
- 39) Stone, M. H.: Boolean Algebras.