

A STUDY OF LOGIC SYNTHESIS AND OPTIMIZATION

BY
A.S.M.MOINUL AHSAN

A THESIS SUBMITTED TO THE DEPARTMENT
OF COMPUTER SCIENCE AND ENGINEERING, BUET, DHAKA,
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE
OF MASTER OF SCIENCE IN ENGINEERING



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DECEMBER, 1994

515.84
1994
MOI

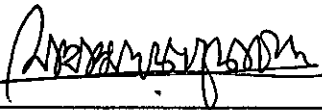
A STUDY OF LOGIC SYNTHESIS AND OPTIMIZATION

A thesis submitted by

A.S.M. MOINUL AHSAN


Roll no. 901818P, Registration no. 84167,
for the partial fulfillment of the degree of
M.Sc. Engg. in Computer Science and Engineering.
Examination held on December 26, 1994.

Approved as to style and contents by:

 26/12/94

DR. MOHAMMAD KAYKOBAD
Associate Professor
Department of Computer Science & Engineering
BUET, Dhaka-1000, Bangladesh

Chairman
(Supervisor)

 26-12-94


DR. MD. SHAMSUL ALAM
Professor and Head
Department of Computer Science & Engineering
BUET, Dhaka-1000, Bangladesh

Member
(Ex officio)



MD. ABDUS SATTAR
Assistant Professor
Department of Computer Science & Engineering
BUET, Dhaka-1000, Bangladesh

Member

 26.12.94

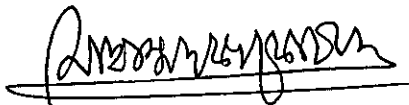
DR. MD. ABDUL MOTTALIB
Associate Professor
Department of Computer Science
Dhaka University
Dhaka-1000, Bangladesh

Member
(External)

CERTIFICATE

This is to certify that the work presented in this thesis is done by me under the supervision of Dr. Mohammad Kaykobad, Associate Professor, Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka. It is also certified that neither this thesis nor any part thereof has been submitted elsewhere for the award of any degree or diploma.

Countersigned by the supervisor



Dr. Mohammad Kaykobad

Signature of the candidate



A.S.M. Moinul Ahsan

ACKNOWLEDGEMENTS

The author would like to express his sincere gratitude and profound indebtedness to his supervisor Dr. Mohammad Kaykobad, Associate Professor, Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET), Dhaka, for his constant guidance, helpful advice, invaluable assistance and endless patience throughout the progress of this work, without which the work could not have been completed.

The author gratefully acknowledge the kind support and encouragement extended to him by Dr. Md. Shamsul Alam, Professor and Head, Department of Computer Science and Engineering, BUET, Dhaka, during the course of this work.

The author conveys his sincere thanks to Md. Abdus Sattar, Assistant Professor, Department of Computer Science and Engineering, BUET, Dhaka, for his kind cooperation during the progress of this study. The author also expresses his sincere gratitude to Dr. Md. Abdul Mottalib, Associate Professor, Department of Computer Science, Dhaka University, Dhaka, for his kind cooperation during the progress of this study and acting as external examiner.

Finally, the author acknowledges with hearty thanks, the all out cooperation of Latifur Rahman Khan, Mohammad Ohiuddin Mahboob, Md. Sanaul Hoque, all the faculty members and his friends of the Computer Science and Engineering Department, BUET, Dhaka.

ABSTRACT

With larger and larger scale of integration the problem being faced by the technology is that logic circuitry are becoming more and more complicated. It is important to be able to optimize logic circuitry, so that circuitry becomes as simple as possible. In this thesis a set of important logic synthesis and optimization methods have been thoroughly studied theoretically, and relative performances of the three of them have been evaluated by carefully designed experiments. In these experiments we apply different types of functions and observe the time elapsed, number of literals, and number of products. For the same set of data we have applied different orientation of data. This study shows that for Quine-McCluskey and Mlynarovic methods both the number of products and the literals are the same but for EXMIN2 the number of products and the number of literals differs in a small extent. When the number of variables increases all the methods need more time, but in the case of Mlynarovic method the time elapsed is fixed for a particular number of variables whatever the number of inputs may be. Time required for Quine-McCluskey and EXMIN2 varies with number of inputs for the same number of variables. For the same set of data EXMIN2 gives the minimal number of product terms for the Arithmetic and Symmetric function. For randomly generated function Quine-McCluskey method gives fewer number of products than Mlynarovic method but number of literals is less in most of the cases in the later method. When number of inputs is less comparable than the maximum possible inputs then Quine-McCluskey method gives fewer number of products than Mlynarovic method. It is seen that from the point of view of product terms and literals EXMIN2 is better than the others. The thesis also gives an elaborate guideline for future research work to proceed in this direction.

CONTENTS

	Page
INTRODUCTION	1
General	1
Importance of the Study of Digital Logic	3
Historical Perspective	5
Thesis Organization and Objective	6
CHAPTER 1. LITERATURE REVIEW	8
1.1 Introduction	8
1.2 Basic Definitions	10
1.3 Axiomatic Definition of Boolean Algebra	12
1.4 Basic Theorems and Properties of Boolean Algebra	14
1.4.1 Duality	14
1.4.2 Basic Theorems	14
1.5 Some Definition of Boolean Algebra	17
1.5.1 Propositional Operation	17
1.5.2 Boolean Functions	18
1.6 Minimization Techniques	21
1.6.1 Minimization by Boolean Algebra	21
1.6.2 Minimization by Map Method	23
1.6.3 Minimization by Tabulation Method	26
1.6.4 Minimization in ESOP Domain	26
CHAPTER 2. QUINE-McCLUSKEY METHOD	27
2.1 Introduction	27
2.2 Tabulation Method for the Minimization of Boolean Functions	28

	Page
2.2.1 The Binary Representation	30
2.2.2 The Decimal Representation	33
2.3 Algorithm	34
CHAPTER 3. CLASSIFICATION OF AND-EXOR EXPRESSIONS	36
3.1 Introduction	36
3.2 Several Classes of AND-EXOR Expressions	37
3.3 Algebraic Identities for the EXOR-Connective	37
3.4 Expansion Theorem	38
3.4.1 Positive Polarity Reed-Muller Expression(PPRME)	39
3.4.2 Fixed Polarity Reed-Muller Expression(FPRME)	40
3.4.3 Kronecker Expression(KRO)	41
3.4.4 Pseudo Reed-Muller Expression(PSDRME)	42
3.4.5 Pseudo Kronecker Expression(PSDKRO)	44
3.4.6 Generalized Reed-Muller Expression(GRME)	45
3.4.7 Exclusive-or Sum-of-Products Expression(ESOP)	46
3.5 Relations Among the Classes	46
CHAPTER 4. LOGIC SYNTHESIS WITH EXOR GATES	47
4.1 Introduction	47
4.2 Design Method of AND-EXOR Circuits	47
4.3 Simplification of AND-EXOR Expressions	48
4.4 Algorithm	57
CHAPTER 5. MINIMIZATION USING REED-MULLER CANONIC EXPANSION	68
5.1 Introduction	68
5.2 Historical Perspective for Minimization of Reed-Muller Canonic Expansions	69
5.3 Reed-Muller Expansion	70
5.4 Techniques for Minimization of RMC Expansions	72

	Page
5.4.1 Map Simplification of Positive Polarity Expansion	72
5.4.2 Tri-state Map Method	75
5.4.3 Map Folding Techniques	77
5.4.4 Transformation from SOP Domain to ESOP Domain	79
5.5 Mlynarovic's Method for Minimization	85
5.6 Modified Algorithm	91
5.6.1 Algorithm	110
CHAPTER 6. EXPERIMENTS AND RESULTS	112
6.1 Design of Experiments	112
6.2 Quine-McCluskey Method	113
6.3 EXMIN2	114
6.4 Mlynarovic's Method for Minimization	116
6.5 Results	117
6.6 Comparison Among the Minimization Techniques	128
CHAPTER 7. CONCLUSIONS AND RECOMMENDATIONS	129
7.1 Conclusions	129
7.2 Suggestions for Further Study	133
APPENDIX A. DATA AND SOP/ESOP/FPRME EXPRESSIONS	134
BIBLIOGRAPHY	148

INTRODUCTION

General:

Digital simply means that information is represented by signals that take on a limited number of discrete values and is processed by devices that normally function only in a limited number of discrete states. The lack of practical devices capable of functioning reliably in more than two discrete states has resulted in the vast majority of digital devices being binary [7], i.e. having signals and states limited to two discrete values. Any structure of physical devices assembled to process or transmit digital information may be termed a digital system.

Logic circuits for digital system may be combinational or sequential. A combinational circuit consists of logic gates whose outputs at any time are determined directly from the present combination of inputs without regard to previous inputs. A combinational circuit performs a specific information-processing operation fully specified logically by a set of Boolean functions.

Sequential circuits employ memory elements (binary cells) in addition to logic gates. Their outputs are a function of the inputs and the state of the memory elements. The state of memory elements, in turn, is a function of previous inputs. As a consequence, the outputs of a sequential circuit depend not only on present inputs, but also on past inputs.

The characteristics of digital systems vary and the approach to their design sometimes varies as well. The information that enters and leaves a digital system are of two categories: 1) Information to be processed or transmitted and 2) Control information. Information in the first category usually occurs in the form of a time-sequence of

information vectors. A vector might be a byte, 8 binary bits; it might be a word of 16 to 64 bits; or it might be several words. The second category, control information, usually occurs in smaller quantities. It is information that guides the digital system in performing its functions. Sometimes, the control information is received only. In other cases, control pulses are sent out to control the information of some other equipments.

Certain digital systems handle only control information. The controller for an elevator is a good example. System of this type may be designed as sequential circuits.

Every digital system is likely to have combinational circuits, so the synthesis optimization of combinational logic design plays an important role in digital logic synthesis and optimization. The procedure of combinational logic design involves following steps: The problem is stated; The number of available input variables and required output variables is determined; The input and output variables are assigned letter symbols; The truth table that defines the required relationships between inputs and outputs are derived; The simplified Boolean function for each output is obtained; The logic diagram is drawn.

The output Boolean functions from the truth table are simplified by any available method, such as algebraic manipulation, the map method or the tabulation procedure. In any particular application, certain restrictions, limitations and criteria will serve as guideline in the process of choosing a particular algebraic expression. Practical design method would have to consider such constraints as (1) minimum number of gates, (2) minimum number of inputs to a gate, (3) minimum propagation time of the signal through the circuit, (4) minimum number of interconnections, (5) limitations of the driving capabilities of each gate. All these criteria cannot be satisfied simultaneously, and since the importance of each constraint is dictated by the particular application, it is difficult to make a general statement as to what constitutes an acceptable simplification. In most cases the simplification begins by satisfying an elementary objective, such as producing a simplified Boolean function in a standard form and from that proceeds to meet any other

performance criteria.

Importance of the Study of Digital Logic:

In the application of control, communication and computer, analog electronic circuits were being used in the earlier half of this century. But for reasons like speed, reliability etc, digital circuitry is being preferred nowadays.

Digital circuits are being used in such diverse fields as information storage, medical instrumentation, process control, calculators and computers, air traffic and digital communications, and in voice and tone synthesis. Digital techniques are also penetrating into areas that were traditionally solved with analog techniques. An example will clarify the point. If it is desired to send analog information from point A to point B, where the value of an electric potential is made proportional to a measured analog quantity. The transmission medium may be a cable or a modulated radio-frequency carrier. In either case, electrical interference signals are present that add to the information signal modifying its content. Suppose that an accuracy of 1 part in 10^4 is required in the information signal. The interference must be kept below that value, or special techniques must be used to remove at the receiving end the unwanted signal component. On the other hand, if a signal is being transmitted as a pulse train which, via its sequence of present or absent pulses in the train, represents information in binary form. It is evident that this communication link can be more noise immune since no admixture of noise will change the information content unless its energy becomes comparable to the energy contained in the information pulse.

Higher power efficiencies of digital systems provide an additional reason for their applications in areas formerly dominated by analog techniques. Consider a transistor used as a switch in a digital circuit, the power dissipated in it is $P_D = I \text{ amperes} \times V \text{ volts}$. When the transistor switch is in the OFF state, I and hence P_D approaches zero. Similarly,

when the transistor switch is in the ON state, V and hence P_D also approaches zero. Power is thus dissipated only during transitions between the ON and OFF states and in the load when the transistor is in the ON state. In contrast, in analog applications both I and V have non-zero values and the transistor dissipates power continuously.

Concurrent with the reduction in size and power consumption there has been a considerable increase in the operating speed of the new semiconductor devices. It is sometimes difficult to fully appreciate the revolutionary changes that were made possible through the invention of the transistor and the subsequent development of integrated circuits.

For example, the largest early computers occupied a volume of hundreds of cubic meters and required many tens of kilowatts of electrical power and a sizeable air-conditioning installation to allow this amount of energy to be dissipated without raising the room temperature to unbearable values. The comparable computational capacity is obtained today in a desk-size computer dissipating one hundredth of the former's power [2].

Digital computer is the best-known example of a digital system. Other examples involve telephone switching exchanges, digital voltmeters, frequency counters and teletype machines. Digital computers have made possible many scientific, industrial and commercial advances that would have been unattainable otherwise. Space program would have been impossible without real time, continuous computer monitoring and many business enterprises function efficiently only with the aid of automatic data processing. The most striking property of a digital computer is its generality [9]. It can follow a sequence of instructions, called a program, that operates on given data. The user can specify and change programs and/or data according to the specific need. As a result of this flexibility, general-purpose digital computers can perform a wide variety of information processing tasks. It is possible to introduce error-detecting and/or error-correcting capabilities in digital system by the inclusion of so redundant information in terms of

binary digits(bits). It is then possible to detect if there is any error in the information. It is even possible to know in which position the error has occurred and it can be easily corrected.

Because of these inherent advantages, various studies were made on different aspects of digital hardware. Beginning from fifties, one finds the emergence of various techniques of minimization of digital logic circuits with the objective of reducing the cost of realization of a digital system. Recently automatic logic synthesis tools are extensively used in VLSI design. Increasing complexity of LSIs has made human design of bug-free logic circuits very difficult. Thus, various automatic logic synthesis tools have become indispensable in LSI design. Most of the logic synthesis tools use the design theory for the circuits consisting of AND, OR and NOT gates. As for control circuits, these tools produce good circuits comparable to the human design. However, they are not so good at the design for arithmetic circuits, error correcting circuits and circuits for telecommunication. Such circuits can be simplified when EXOR gates are effectively used. Therefore, in order to develop a logic synthesis tools for such circuits, a design theory utilizing EXOR gates is very important [29].

Historical Perspective:

Modern digital computers are designed and maintained, and their operation is analyzed, by using techniques and symbology from a field of mathematics called modern algebra. Algebraists have studied for over a hundred years mathematical systems called Boolean algebra. The name Boolean algebra honors a fascinating English mathematician, George Boole, who in 1854 published a classic book "An Investigation of the Laws of Thought, on Which Are Founded the Mathematical Theories of Logic and Probabilities". This theory based on the Aristotelian logic concepts of "true" (represented by a binary 1) and "false" (represented by a binary 0) found a practical application 84 years later when Shannon introduced a two-valued Boolean algebra called switching algebra, in which he

demonstrated that the properties of bistable electrical switching circuits can be represented by this algebra (see in [21]). Boolean algebras can be interpreted in terms of sets and logical propositions is discussed in details by Whitesitt (see in [30]).

In the late 1930's switching devices consisted mainly of relays since the vacuum tube was not deemed sufficiently reliable to serve as a component in large switching networks, such as telephone systems. The successful application of Boolean algebra to the analysis of switching networks may be viewed as one of the information breakthroughs in processing digital information. Much emphasis was placed then on minimizing the number of components utilized in a switching network.

Thesis Organization and Objective:

This thesis comprises seven chapters. Chapter one is the literature review. In this chapter Boolean Algebra and its relation with the switching algebra is briefly discussed and some definitions related to Boolean function minimization technique are mentioned. Besides, different techniques for minimization of a Boolean function are also briefly discussed, which is a fundamental step towards detailed study of the optimization tools used.

Chapter two discusses Quine-McCluskey method. It has been found that the Quine-McCluskey method is the most popular method for any number of variables and suitable for programming on a digital computer for systematic minimization of large Boolean functions. In this thesis work, Quine-McCluskey method is implemented in a different approach for the ease of software implementation but the basic theme is the same. This method is included in this thesis for the comparison of number of products, literals and time with that of EXOR expression for the same function. This would reveal the relative advantages and disadvantages of SOP and ESOP expressions for the same function.

Chapter three discusses different types of EXOR expressions as classified by T. Sasao. Relations between these classes are also discussed. This classification is a strong tool for the study of EXOR minimization techniques.

Chapter four discusses the EXMIN2 Algorithm, which gives the EXOR expression in ESOP form. ESOP gives the minimal number of products among the different EXOR expressions.

Chapter five discusses fixed polarity Reed-Muller expression, which is a subset of ESOP. Algorithm used is known as Mlynarovic's method which is a mathematical model for the minimization of Boolean functions into EXOR expressions. This algorithm is included in this thesis for a comparative study with the ESOP expression for the same function. In this chapter different techniques for the minimization of Reed-Muller expression is also briefly discussed for further studies.

Chapter six presents the experimental results based on the algorithms discussed so far in the previous chapters. This chapter compares different minimization techniques based on experimental results. In chapter seven we make conclusion of our findings and recommend some issues for further research in this direction.

This thesis is an attempt to analyze the performance of different logic minimization techniques. Mainly we have concentrated our study on time, number of literals, number of products, the permutation of the inputs and the effect of different kinds of functions. We have designed simulation programs to foresee the response of all these minimization techniques and it is hoped that these programs would be useful for any number of variables. All the simulation experiments have been carried out on an IBM PC compatible machine having 80386-SX processor of 33 Mhz speed.

CHAPTER 1

LITERATURE REVIEW

1.1 Introduction:

Logic networks may be analyzed and designed to a large extent with little or no knowledge of electronics, hydraulics or other technologies. However, if design is to lead to optimized operational hardware, then the logic designer should must respect the technology to be used to realize a paper design. Limitations and imperfections of technology must be overcome by logic design. A lot of investigation has been done on design and minimization of digital circuits in the last two decades [13]. The normal form of a Boolean function can be considered to be a measure of the complexity of the corresponding logic circuit, the minimal form ensuring in general, minimal complexity of the corresponding logic circuit [31].

Minimization of SOP's has been studied for more than 30 years. Various algorithms have been developed to obtain minimum and near minimum PLA's [23]. In the evaluation of Boolean minimization algorithms, one of the important criteria is the degree of minimality obtained. The degree of minimality, however, can be determined only by minimizing examples for which a minimal solution is known. As the number of input variables increases, it becomes increasingly difficult to find examples for which this minimum is known. So, many attempts have been made to increase the size of problems that can be minimized by sacrificing absolute minimality or modifying the cost function.

According to Dietmeyer [4] minimization is the process of obtaining that expression of a switching function which is optimum under some criterion. Usually a cost criterion is

established and the optimum expression is the one which dictates a minimum cost realization of a function. There are various criteria to determine minimal cost; the most common are: minimum number of appearances of literals; minimum number of literals in a sum of products expression; minimum number of terms in a sum of products expression, provided there is no other such expression with the same number of terms and with fewer literals. These in turn are total gate count, number of interconnections, gate type and so on. Interconnections have capacitances that result in added propagation delays and they contribute undesirable coupling between signals, necessitating "noise margins", and can be a source of signal loss.

The type of circuits used when actually constructing a network influences the cost criterion and the equation to circuit transformation must be clearly established. Cost criteria can be simple or very complex. At one time diodes were expensive and reducing the number of diodes required in AND and OR gates was very worthwhile. Then transistors in NAND and NOR gates became the expensive items, it was important to minimize transistor count, ignoring diode, resistor and fabrication cost. For the designer who interconnects available integrated circuits, IC count is a meaningful criterion for purpose of comparing alternative realizations of a switching function. Integrated circuit cost making a connection to a gate or mounting the gate on supporting material can equal or exceed the cost of gate itself. An ordinary programmable logic array (PLA) has an AND-OR structure. Because PLA's can be designed automatically, easily tested and easily modified, they are extensively used in modern LSI's [23]. By replacing the OR array with the EXOR array, we can have AND-EXOR PLA which have several advantages over AND-OR PLA's. But EXOR's are more expensive and slower than OR's and design of AND-EXOR PLA's is more difficult than that of AND-OR PLA's.

So it is evident that the process of minimization and optimization of Boolean functions help a designer to realize an elegant logic network.

The axioms and theorems of Boolean algebra can be used to minimize a Boolean function. There should be an orderly procedure in applying these axioms and theorems. Several orderly procedures (Algorithms) have been developed which will give a minimal sum of products/EXOR sum of products of Boolean functions. In this chapter we will describe briefly some definitions and techniques related to Boolean function minimization techniques.

1.2 Basic Definitions:

Boolean algebra, like any other deductive mathematical system, may be defined with a set of elements, a set of operators and a number of unproved axioms or postulates. A set of elements is any collection of objects having a common property. A set with x, y elements is specified as $A = \{x, y\}$. If S is a set and x and y are certain objects, then $x \in S$ denotes that x is a member of the set S and $y \notin S$ denotes that y is not an element of S . A binary operator defined on a set S of elements is a rule that assigns to each pair of elements from S a unique element from S . As an example, consider the relation $a * b = c$. We say that $*$ is a binary operation if it specifies a rule for finding c from the pair (a, b) and also if $a, b, c \in S$. However, $*$ is not a binary operator if $a, b \in S$, while the rule finds $c \notin S$.

The postulates of a mathematical system form the basic assumptions from which it is possible to deduce the rules, theorems and properties of the system. The most common postulates used to formulate various algebraic structures are:

1) **Closure:** A set S is closed with respect to a binary operator if, for every pair of elements of S , the binary operator specifies a rule for obtaining a unique element of S . For example, the set of natural numbers $N = \{1, 2, 3, 4, \dots\}$ is closed w.r.t the binary operator plus(+) by the rules of arithmetic addition, since for any $a, b \in N$ we obtain a unique $c \in N$ by the operation $a + b = c$.

2) **Associative law:** A binary operator $*$ on a set S is said to be associative whenever:

$$(x*y)*z=x*(y*z) \text{ for all } x,y,z \in S.$$

3) **Commutative law:** A binary operator $*$ on a set S is said to be commutative whenever:

$$x*y=y*x \text{ for all } x,y \in S.$$

4) **Identity element:** A set S is said to have an identity element with respect to a binary operation $*$ on S if there exists an element $e \in S$ with the property:

$$e*x=x*e=x \text{ for every } x \in S.$$

5) **Inverse:** A set S having the identity element e with respect to a binary operator $*$ is said to have an inverse whenever, for every $x \in S$, there exists an element $y \in S$ such that

$$x*y=e$$

6) **distributive law:** if $*$ and $.$ are two binary operators on a set S , $*$ is said to be distributive over $.$ whenever:

$$x*(y.z)=(x*y).(x*z)$$

A field is a set of elements, together with two binary operators, each having properties 1 to 5 and both operators combined to give property 6. The set of real numbers together with the binary operators $+$ and $.$ form the field of real numbers. The field of real numbers is the basis for arithmetic and ordinary algebra. The operators and postulates have the following meanings.

The binary operator $+$ define addition.

The additive identity is 0.

The additive inverse defines subtraction.

The binary operator $.$ defines multiplication.

The multiplicative identity is 1.

The multiplicative inverse of $a=1/a$ defines division, i.e., $a.1/a=1$.

The only distributive law applicable is that of $.$ over $+$:

$$a.(b+c)=(a.b)+(a.c)$$

1.3 Axiomatic Definition of Boolean Algebra:

Boolean algebra is an algebraic structure defined on a set of elements B together with two binary operator $+$ and \cdot provided the following Huntington postulates are satisfied:

1.a) Closure w.r.t the operator $+$

b) Closure w.r.t the operator \cdot

2.a) An identity element w.r.t $+$, designated by 0:

$$x+0 = 0+x = x$$

b) An identity element w.r.t \cdot , designated by 1:

$$x \cdot 1 = 1 \cdot x = x$$

3.a) Commutative w.r.t $+$:

$$x+y = y+x$$

b) Commutative w.r.t \cdot :

$$x \cdot y = y \cdot x$$

4.a) \cdot is distributive over $+$:

$$x \cdot (y+z) = (x \cdot y) + (x \cdot z)$$

b) $+$ is distributive over \cdot :

$$x + (y \cdot z) = (x+y) \cdot (x+z)$$

5. For every element $x \in B$, there exist an element $\bar{x} \in B$ such that:

a) $x + \bar{x} = 1$

b) $x \cdot \bar{x} = 0$

6. There exist at least two elements $x, y \in B$ such that

$$x \neq y$$

Comparison of Boolean algebra with arithmetic and ordinary algebra:

1) Huntington postulates do not include the associative law. However, this law holds for Boolean algebra.

2) the distributive law of $+$ over \cdot is valid for Boolean algebra, but not for ordinary

algebra.

3) Boolean algebra does not have additive or multiplicative inverses; therefore, there are no subtraction or division operations.

4) Postulate 5 defines an operator called complement which is not available in ordinary algebra.

5) Ordinary algebra deals with the real numbers, which constitute an infinite set of elements. Boolean algebra deals with the as yet undefined set of elements B, but in the two-valued Boolean algebra B is defined as a set with only two elements, 0 and 1.

In order to have a Boolean algebra, one must show:

1. The elements of the set B.
2. The rules of operation for the two binary operators.
3. That the set of elements B, together with the two operators, satisfies the six Huntington postulates.

TWO-VALUED Boolean algebra:

Two-valued Boolean algebra has applications in set theory and in propositional logic. A two-valued Boolean algebra is defined on a set of two elements, $B=\{0,1\}$, with rules for the two binary operators $+$ and \cdot as shown in the following operator tables.

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

x	y	$x+y$
0	0	0
0	1	1
1	0	1
1	1	1

x	\bar{x}
0	1
1	0

It can be easily verified that Huntington postulates are valid for the set $B=\{0,1\}$ and the two binary operators defined above.

Thus a two-valued Boolean algebra having a set of two elements, 1 and 0, two binary operators with operation rules equivalent to the AND and OR operations and a complement operator equivalent to the NOT operator has been established and defined in a formal mathematical manner. The two-valued Boolean algebra defined in this section is also called "switching algebra" by engineers. From here on, we shall drop adjective "two-valued" from Boolean algebra in the subsequent discussion.

1.4 Basic Theorems and Properties of Boolean Algebra:

1.4.1 Duality:

The Huntington postulates have been listed in pairs and designated by part(a) and part(b). One part may be obtained from the other if the binary operator and the identity elements are interchanged. This important property of Boolean algebra is called the duality principle. It states that every algebraic expression deducible from the postulates of Boolean algebra remains valid if the operators and identity elements are interchanged.

1.4.2 Basic Theorems:

Boolean algebra is a closed system consisting of a finite set S of two or more elements, subject to an equivalence relation(=) and the three binary operators OR, AND, and NOT, such that for every element x and y in the set, the operations $x+y$, $x.y$, \bar{x} , and \bar{y} are also uniquely defined in the set, and the Huntington's postulates are satisfied. These postulates and their duals are listed as follows:

Postulate 1 : For each operation there exists unique elements, 1 and 0, in set S such that for x in S ,

$$\text{a) } x + 0 = x \quad \text{and} \quad x.1 = x$$

and

$$\text{b) } x.0 = 0 \quad \text{and} \quad x + 1 = 1$$

Postulate 2 : The operations are commutative for every x and y in set S such that

$$\text{a) } x + y = y + x \quad \text{and} \quad \text{b) } x.y = y.x$$

Postulate 3 : The operations are distributive. For all x , y , and z in set S ,

$$\text{a) } x.(y + z) = (x.y) + (x.z)$$

and

$$\text{b) } x + (y.z) = (x + y).(x + z)$$

Postulate 4 : The operations are associative. For every x , y , and z in set S ,

$$\text{a) } x + (y + z) = (x + y) + z$$

and

$$\text{b) } x.(y.z) = (x.y).z$$

Postulate 5 : For every element x in the set S there exists an element \bar{x} such that

$$\text{a) } x + \bar{x} = 1 \quad \text{and} \quad \text{b) } x.\bar{x} = 0$$

Boolean theorems, whose primary application is in the minimization of logic circuits, can now be derived using the stated postulates. The theorems are as follows:

Theorem 1 : The Law of Idempotency. For all x in set S ,

$$\text{a) } x + x = x \quad \text{and} \quad \text{b) } x.x = x$$

Theorem 2 : The Law of Absorption. For all x and y in set S ,

$$\text{a) } x + (x.y) = x \quad \text{and} \quad \text{b) } x.(x + y) = x$$

Theorem 3 : The Law of Identity. For all x and y in set S , if

$$\text{a) } x + y = y \quad \text{and} \quad \text{b) } x.y = y \quad \text{then } x = y$$

Theorem 4 : The Law of Complements. For all x and y in set S , if

$$\text{a) } x + y = 1 \quad \text{and} \quad \text{b) } x.y = 0 \quad \text{then } x = \bar{y}$$

Theorem 5 : The Law of Involution. For all x in set S ,

$$x = \bar{\bar{x}}$$

Theorem 6 : DeMorgan's Law. For all x, y, \dots , and z in set S

$$\text{a) } \overline{x.y.\dots.z} = \bar{x} + \bar{y} + \dots + \bar{z}$$

and

$$b) \overline{x + y + \dots + z} = \bar{x} \cdot \bar{y} \cdot \dots \cdot \bar{z}$$

This theorem implies that any function can be complemented by changing the ORs to ANDs, ANDs to ORs, and complementing each of the variables.

Theorem 7 : The Law of Elimination. For all x and y in set S ,

$$a) x + \bar{x} \cdot y = x + y \quad \text{and} \quad b) x \cdot (\bar{x} + y) = x \cdot y$$

Theorem 8 : The Law of Consensus. For all x, y , and z in set S ,

$$a) x \cdot y + \bar{x} \cdot z + y \cdot z = x \cdot y + \bar{x} \cdot z$$

and

$$b) (x + y) \cdot (\bar{x} + z) \cdot (y + z) = (x + y) \cdot (\bar{x} + z)$$

Theorem 9 : The Law of Interchange. For all x, y , and z in set S ,

$$a) (x \cdot y) + (\bar{x} \cdot z) = (x + z) \cdot (\bar{x} + y)$$

and

$$b) (x + y) \cdot (\bar{x} + z) = (x \cdot z) + (\bar{x} \cdot y)$$

Theorem 10 : The Generalized Functional Laws. The AND/OR operation of a variable X and a multi-variable composite function that is also a function of X is equivalent to similar AND/OR operation of X with the composite function whose X is replaced by 0:

$$a) X + f(X, Y, \dots, Z) = X + f(0, Y, \dots, Z)$$

$$b) X \cdot f(X, Y, \dots, Z) = X \cdot f(0, Y, \dots, Z)$$

For all X, Y, \dots , and Z in set S ,

$$a) f(X, Y, \dots, Z) = X \cdot f(1, Y, \dots, Z) + \bar{X} \cdot f(0, Y, \dots, Z)$$

$$b) f(X, Y, \dots, Z) = [X + f(0, Y, \dots, Z)] \cdot [\bar{X} + f(1, Y, \dots, Z)]$$

Operator Precedence:

The operator precedence for evaluating Boolean expressions is

(1) Parentheses (2) NOT (3) AND and (4) OR.

1.5 Some Definition of Boolean Algebra:

1.5.1 Propositional Operation: (AND, OR, NOT)

Another interpretation of a Boolean algebra with two elements in the set B is in terms of propositional logic. A proposition in this context is the content or meaning of a declarative sentence for which it is possible to determine the truth or falsity. Thus, a sentence must be free of ambiguity and must not be self-contradictory in order to qualify as a proposition. The following statements are all propositions:

- a) The moon is made of green cheese.
- b) 4 is a prime number.
- c) It snowed on the island that is now called Manhattan on the day the king of England signed the Magna Carta.

of the above propositions, a) and b) are false and c) may or may not be true. The statement "All men are tall" is not a proposition, since it is ambiguous.

Proposition can be combined or manipulated by operations AND, OR, NOT, etc. The AND combination of two propositions is true if both propositions are true and is denoted by (.) or (\wedge). If either or both of them are false, the combination is false. This new proposition is called conjunction of the two propositions.

The OR combination of two propositions is true when either or both of them are true and is false if both of them are false. This is called the logical sum or disjunction, and is denoted by plus(+) or (\vee) sign between the two symbols.

Corresponding to any given proposition P, it is possible to form another proposition which asserts that P is false. This new proposition is called the denial or complement of P and is written as P' or $\sim P$. If P is the statement "It is raining" then P' is the statement "It is false that it is raining" or "It is not raining".

1.5.2 Boolean Functions:

A Boolean function is an expression formed with binary variables, the two binary operators OR and AND, the unary operator NOT, parentheses, and equal sign. For a given value of the variables, the function can be either 0 or 1. For example, consider the Boolean function $F_1 = x \cdot y \cdot \bar{z}$

The function F_1 is equal to 1 if $x=1$ and $y=1$ and $\bar{z} = 1$; otherwise $F_1=0$. The above expression is an algebraic expression of a Boolean function.

Truth table:

A Boolean function may also be represented in a truth table. For a n binary variable input function, there are 2^n input combinations. A table can be prepared representing the value of function for all 2^n possible combinations. This table is known as truth table of the Boolean function. The algebraic expression of a Boolean function is not unique, whereas the truth table of a Boolean function is unique.

Literals:

A literal is a primed or unprimed variable. When a Boolean function is implemented with logic gates, each literal in the function designates an input to a gate, and each term is implemented with a gate. The minimization of the number of literals and the number of terms results in a circuit with fewer gates and interconnections.

Minterms and Maxterms:

A binary variable may appear either in its normal form (x) or in its complement form (\bar{x}). If two binary variables x and y combined with an AND operation, there are four possible combinations: $\bar{x} \cdot \bar{y}$, $\bar{x} \cdot y$, $x \cdot \bar{y}$ and $x \cdot y$. Each of the four terms is called minterms or a standard product. In a similar manner, n variables can be combined to form 2^n minterms. In a similar fashion, n variables forming an OR term, with each variable being primed or unprimed, provide 2^n possible combinations, called maxterms or standard

sums.

Canonical form:

A Boolean function may be expressed algebraically from a given truth table by forming a minterm for each combination of the variables which produces a "1" in the function and then taking the OR of all those terms. Similarly any Boolean function can be expressed as a product of maxterms by forming a maxterm for each combination of the variables which produces a "0" in the function and then form the AND of all those maxterms. Boolean function expressed as a sum of minterms or product of maxterms are said to be in canonical form.

Standard form:

In this configuration, the terms that form the function may contain one, two or any number of literals. There are two types of standard forms: SOP(sum of products) and POS(product of sum).

An example of a function expressed in SOP forms is:

$$F_1 = \bar{y} + x \cdot y + \bar{x} \cdot y \cdot \bar{z}$$

An example of a function expressed in POS forms is:

$$F_2 = x(\bar{y} + z)(\bar{x} + y + \bar{z} + w)$$

Prime Implicant:

A prime implicant P, is said to cover a set of standard product terms.

Essential prime implicant:

The term essential prime implicant denotes a prime implicant that covers at least one standard product term that cannot be covered by any other prime implicant. The essential prime implicants, therefore, must be included to get a minimum cover of the function.

Partially symmetric function:

A function $f(x_1, x_2, \dots, x_n)$ is partially symmetric in $x_{i_1}, x_{i_2}, \dots, x_{i_m}$ iff it is unchanged by any permutation of the variables $x_{i_1}, x_{i_2}, \dots, x_{i_m}$.

Symmetric function:

A function $f(x_1, x_2, \dots, x_n)$ is totally symmetric iff it is unchanged by any permutation of its variables.

Positive function:

A function $f(x_1, x_2, \dots, x_n)$ is positive in x_i iff it is possible to write a sum-of-products expression for f in which \bar{x}_i does not appear.

Vacuous function:

A function $f(x_1, x_2, \dots, x_n)$ is vacuous in x_i iff it is possible to write a sum-of-products expression for f in which neither x_i nor \bar{x}_i appear.

Unate function:

A function $f(x_1, x_2, \dots, x_n)$ is unate function iff it is positive or negative in each of its variables.

Parity function:

A function is called parity(odd) function if its value is 1 when odd number of inputs are 1 or a function is called parity(even) function if its value is 1 when even number of inputs are 1.

Majority function:

A function is called majority function if its value is 1 when more than half of its input variables are 1.

1.6 Minimization Technique:

The complexity of digital logic gates that implement a Boolean function is directly related to the complexity of the algebraic expression from which the function is implemented. Although the truth table representation of a function is unique, expressed algebraically, it can appear in many different forms. There are different methods to minimize a Boolean function, some of these are described briefly as follows:

1.6.1 Minimization by Boolean Algebra:

In this techniques canonical form of a Boolean function is converted to minimal form by applying the different postulates and theorems of Boolean algebra.

Example: Simplify $(x + y)(x + \bar{x}.y).z + \bar{x}.y + x.y.z + \overline{x(y + z)}$

Solution: $(x + y)(x + \bar{x}.y).z + \bar{x}.y + x.y.z + \overline{x(y + z)}$

$$= (x + y)(x + y).z + \bar{x}.y + x.y.z + \overline{x(y + z)} \quad [\text{Th.7(a)}]$$

$$= (x + x.y + x.y + 0).z + \bar{x}.y + x.y.z + \overline{x(y + z)} \quad [\text{Th.1(b)}]$$

$$= x(1 + y + y).z + \bar{x}.y + x.y.z + \overline{x(y + z)}$$

$$= x.z + \bar{x}.y + x.y.z + \overline{x(y + z)} \quad [\text{P.5(a)}]$$

$$= x \cdot z(1 + y) + \bar{x} \cdot y + \overline{x(\bar{y} + \bar{z})}$$

$$= x \cdot z + \bar{x} \cdot y + \overline{x(\bar{y} + \bar{z})}$$

$$= x \cdot z + \bar{x} \cdot y + \bar{x} + \overline{\bar{y} + \bar{z}} \quad [\text{Th.6(a)}]$$

$$= x \cdot z + \bar{x} \cdot y + \bar{x} + y \cdot z \quad [\text{Th.6(b)}]$$

$$= x \cdot z + \bar{x} \cdot y + \bar{x} \quad [\text{Th.8(a)}]$$

$$= x \cdot z + \bar{x}(y + 1)$$

$$= x \cdot z + \bar{x} \quad [\text{P.1(b)}]$$

$$= z + \bar{x} \quad [\text{Th.7(a)}]$$

However this procedure of minimization is awkward because it lacks specific rules to predict each succeeding step in the manipulative process and is ineffective for expressions of even a small number (e.g. four or five) of variables. Consider the minimization of the function $f(x,y,z)$ below

$$f(x, y, z) = \bar{x} \cdot y \cdot \bar{z} + \bar{x} \cdot \bar{y} \cdot \bar{z} + x \cdot \bar{y} \cdot \bar{z} + \bar{x} \cdot y \cdot z + x \cdot y \cdot z + x \cdot \bar{y} \cdot z$$

The combination of the first and second, second and third, fourth and fifth, fifth and sixth terms yield a reduced expression

$$f(x, y, z) = \bar{x} \cdot \bar{z} + \bar{y} \cdot \bar{z} + y \cdot z + x \cdot z$$

This expression is said to be in an irredundant form, since any attempt to reduce it, either by deleting any of the four terms or by removing a literal, will yield an expression which is not equivalent to f .

The above reduction procedure is not unique and a different combination of terms may yield different reduced expressions. In fact, if we combine the first and second, the third and sixth, the fourth and fifth terms of f , we obtain the expression

$$f(x, y, z) = \bar{x} \cdot \bar{z} + x \cdot \bar{y} + y \cdot z$$

In a similar manner, by combining the first and fourth terms, the second and third, the fifth and sixth, we obtain a third irredundant expression,

$$f(x, y, z) = \bar{x} \cdot y + \bar{y} \cdot \bar{z} + x \cdot z$$

while all three expressions are irredundant, only the latter two are minimal. Consequently, an irredundant expression is not necessarily minimal, nor is the minimal expression always unique.

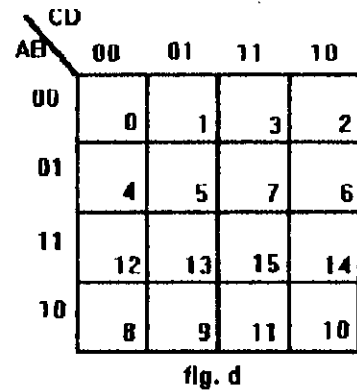
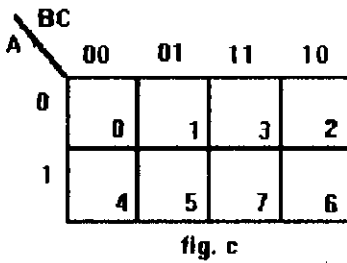
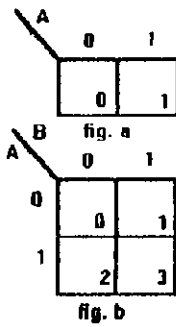
1.6.2 Minimization by Map Method:

The map method provides a simple straight forward procedure for minimizing Boolean functions. This method may be regarded either as a pictorial form of a truth table or as an extension of the Venn diagram. The map method, first proposed by Veitch and slightly modified by karnaugh, is also known as the "Veitch diagram" or the "karnaugh map". The map is a diagram made up of squares. Each square represents one minterm. Each minterm of the standard product Canonical form is arranged in such a fashion so as to provide adjacent places for terms that differ in only one variable. In fact, the map represents a visual diagram of all possible ways a function may be expressed in a standard form.

Formation of k-Map:

The k-Map is a set of 2^n squares arranged in an ordered two dimensional array. Each of the squares on the map corresponds to one of the 2^n possible minterms. The possible k-

Maps for functions up to four variable are shown below



Each of the squares on the map, called a cell, has an entry equal to decimal value of the corresponding minterms. Along the top of the map, the variable values are listed in such a way that as one scans the values from left to right or from right to left, from cell to cell, only one variable changes. Each cell is thus adjacent to the next. The cells on the right end are only one unit distance from the cells on the left. Similarly cells on the top are one unit distance from the cells on the bottom. Entries of 1 are made in the cells corresponding to the minterms that make the function a 1 in SOP form. A 0 is entered in each of the remaining cells.

Minimization process:

Simplification of a Boolean function plotted on a k-map is a process of correctly grouping the adjacent 1s. The rules for forming groups are as follows:

Rule 1: If all the entries in an n-variable k-map are 1s, the group size is 2^n and the corresponding minimized function is given by $f(A,B,\dots) = 1$

Rule 2: For n variables, the largest nontrivial group size is 2^{n-1} . All group sizes are power of 2. For n variables, group sizes of $2^n, 2^{n-1}, 2^{n-2}, \dots, 2^0$ are allowed.

Rule 3: In any group it must be possible to start at a cell and travel from cell to cell, with moves of one unit length, through each cell in the group without passing through any cell twice and return to the starting cell.

Using the rules for forming groups, we now apply the rules to arrive at a minimum SOP function as follows:

1. In the process of grouping, each 1 on the map must be included in at least one group.
2. For n variables, first look for 1s that cannot be grouped with any other 1 and circle these. These isolated 1s will appear as n literal minterms in the final function.
3. Look for 1s that can be grouped with only one other 1 and circle these groups as pairs.
4. Next look for 1s that can be grouped only in groups of four and circle these groups.
5. Continue the grouping process until group sizes of 2^{n-1} have been considered.
6. This grouping process should be stopped at any time that each 1 has been included in at least one group. Next, eliminate the smaller groups, if any, that are enclosed completely within a larger group.
7. For each group, determine which literals remain constant in each cell in the group. These literals when ANDed together form the product terms of SOP function. For x groups, there will be x product terms. For an n -variable function, a group size of 2^k would contribute a product term of $n - k$ literals.

The map method of simplification is convenient as long as the number of variables does not exceed five or six. As the number of variables increases the excessive number of square prevents a reasonable selection of adjacent squares. The obvious disadvantage of the map is that it is essentially a trial-and-error procedure which relies on the ability of the human user to recognize certain patterns. For functions of six or more variables, it is difficult to be sure that the best selection has been made. Further it is not easily programmable for solution by digital computer.

1.6.3 Minimization by Tabulation Method:

It is a specific step-by step procedure that is guaranteed to produce a simplified standard form expression for a function. It can be applied to problems with many variables and has the advantage of being suitable for machine computation. However, it is quite tedious for human use and is prone to mistakes because of its routine, monotonous process. The tabulation method was first formulated by Quine and later improved by McCluskey. The tabular method of simplification consists of the following parts:

- 1) The first is to find by an exhaustive search all the terms that are candidates for inclusion in the simplified function. These terms are called prime-implicants.
- 2) From the set of all prime implicants, a set of essential prime implicants is determined by making a prime implicant chart.
- 3) From the remaining prime implicants a minimum cover is obtained for the remaining minterms.

This method is studied elaborately in the chapter 2.

1.6.4 Minimization in ESOP Domain:

Logic design using exclusive-OR gates offers two advantages when compared to conventional Boolean realizations using AND/OR/NAND/NOR gates. Firstly in certain cases more economic realizations in terms of the number of gates and/or interconnections may be obtained. Secondly, the testability of circuits is significantly improved. There are different techniques to minimize a function into Exclusive sum of products (ESOP) domain. Most of them are of heuristic nature and have their relative advantages and disadvantages. Again, as there are different kinds of EXOR expression, so different techniques have evolved for each classification and there are still to come. Such minimization techniques are illustrated from chapter 3 to 5.

CHAPTER 2

QUINE-McCLUSKEY METHOD

2.1 Introduction:

The manipulation of Boolean functions is one of the most important operations in various applications of computer-aided design of digital systems, such as test generation, synthesis and verification. The efficiency of the Boolean manipulation depends on the form of representation of the Boolean functions [12]. Recently automatic logic synthesis tools are extensively used in VLSI design. Most logic synthesis tools use AND and OR gates as basic logic elements, and they derive multi-level logic circuits from AND-OR two-level circuits. Thus the minimization of sum-of-products expressions (SOPs), which corresponds to the minimization of AND-OR two-level circuits, is vitally important in such tools. The minimization methods should handle Boolean functions containing any number of input variables and have the advantages of being suitable for machine computation. For the minimization of a Boolean function in SOP domain there are map method, tabulation method, N-cube method etc. Among them map and tabulation methods are widely used. The map method of simplification is a trial-and-error procedure which relies on the ability of the human user to recognize certain patterns. For functions of six or more variables, it is difficult to be sure that the best selection has been made. Further it is difficult to implement for machine computation. In this chapter the time complexity of tabulation method would be studied.

2.2 Tabulation Method for the Minimization of Boolean Functions:

It is a specific step-by-step procedure that is guaranteed to produce a simplified standard-form expression for a function. It can be applied to problems with many variables and has the advantage of being suitable for machine computation. However, it is quite tedious for human use and is prone to mistakes because of its routine, monotonous process. The tabulation method was first formulated by Quine and later improved by McCluskey. It is also known as the Quine-McCluskey method. The Quine-McCluskey's (Q-M) tabular reduction technique for single-output function is summarized by the following list of steps.

Step 1:

Convert the minterms to binary form and then group the minterms in accordance with the total number of 1s in their binary representations. The minterms will be referred to as zero-cubes.

Step 2:

Make all possible groupings of two zero-cubes. These groups are known as one-cubes (they have one don't-care in their Boolean cell representation). This step must be followed by possible groupings of pairs that will be referred to as two-cubes (two don't-cares in their Boolean cell representation). This process is continued to produce higher-order cubes until no higher cubes can be formed.

Step 3:

Identify those cubes that couldn't be used in the formation of higher-order cubes. These cubes are called prime-implicants (PIs). The PIs represent all possible groups with no smaller group being part of a larger group.

Step 4:

Identify those PIs that are necessary to provide a minimum covering of all the minterms, that is, each minterm of the original function must be in at least one group. This identification is accomplished by constructing an implication table that, in turn, is used for finding the desired minimum cover. The rules for constructing the implication table are summarized as follows.

1. Use all minterms of the original function as the column headings.
2. Use the PIs as the row labels. It is better to separate the PIs by order. If there are two PIs that cover a minterm, the higher-order PI is preferred.
3. A check mark, X, is entered in each matrix position that corresponds to a minterm covered by a particular PI.
4. Examine the table entries and locate the essential PIs. An essential PI is one that is the only cover for one or more minterms. All minterms that are covered by an essential PI are then removed from active consideration.
5. The secondary essential PIs are determined from the remaining PIs by making judicious choices. A PI is preferred over another if (a) it covers more of the remaining minterms, or (b) it is higher in order than the other. This process is continued until all of the minterms have been considered.

Step 4 can be resolved in the following ways:

- 1) There is a flag, corresponding to each minterm.
- 2) Sort the PIs in the decreasing order of cubes.
- 3) If a minterm is covered by a single PI then the corresponding flag is set to 'P' otherwise it is 'F'.
- 4) Examine the flags of minterms and locate the essential PIs. An essential PI is one that is the only cover for one or more minterms, i.e., minterms have flag value 'P'. All minterms that are covered by an essential PI are then removed from active consideration.

5) Choose the prime implicants which are disjoint with essential prime implicants and then choose the prime implicants sequentially until all the minterms are covered.

2.2.1 The Binary Representation:

Example: Using the Q-M tabular procedure, find a minimum SOP expression for the Boolean function given by

$$f(W, X, Y, Z) = \sum m(0, 2, 3, 5, 7, 8, 10, 13, 15)$$

Solution:

Step 1: The minterms are classified as shown below:

Minterms	Binary Representation	Number of 1s	Check
0	0000	0	x
2	0010	1	x
8	1000	1	x
3	0011	2	x
5	0101	2	x
10	1010	2	x
7	0111	3	x
13	1101	3	x
15	1111	4	x

Step 2: Next we locate the one-cubes. Two minterms may be considered for combination only if the number of 1s present in their binary representation differs by one. Moreover, the minterms must differ by a power of 2 to form valid group, i.e., two minterms would be combined if the binary representations of the two differ in one position. Starting with the top section, we can match 0 and 2 because they differ in the second position from the

right. Similarly 0 and 8 can be combined because they differ in the fourth position from the right. Comparing in this way we get the one-cube chart shown below.

One-cubes	Binary Representation	Number of 1s	Check
0,2	00-0	0	x
0,8	-000	0	x
2,3	001-	1	x
2,10	-010	1	x
8,10	10-0	1	x
3,7	0-11	2	x
5,7	01-1	2	x
5,13	-101	2	x
7,15	-111	3	x
13,15	11-1	3	x

In all these cases the difference between the minterm is always a positive power of 2. Any time a group is formed, a check mark, x, is placed in the check column. This is continued until no more one-cubes can be formed.

Next we proceed to determine the two-cubes from the already obtained one-cubes. A proper two-cube may be located by combining two one-cubes if the two differ in one position. The two-cube chart shown below

Two-cubes	Binary Representation	Number of 1s	Check
0,2,8,10	-0-0	0	
0,2,8,10	-0-0	0	
5,7,13,15	-1-1	1	
5,7,13,15	-1-1	1	

It is seen that for every two-cube, there will always be two pairs of one-cubes that will produce the same two-cube. Both of these pairs, therefore, are checked off. So discarding the duplicate cubes we have the two-cube chart

Two-cubes	Binary Representation	Number of 1s	Check
0,2,8,10	-0-0	0	
5,7,13,15	-1-1	1	

The two-cubes are compared in similar fashion. In this example no comparison can be made since the comparisons must be made between adjacent two-cube groups.

Step 3: All cubes without x's are called PIs. The PI list for this function is obtained as follows

- a) 0,2,8,10
- b) 5,7,13,15
- c) 2,3
- d) 3,7

Step 4: sorting the PIs in the decreasing order of cubes we have the following PI chart along with minterms.

PI	0	2	3	5	7	8	10	13	15	Check
0,2,8,10	x	x				x	x			*
5,7,13,15				x	x			x	x	*
2,3		x	x							
3,7			x		x					

From the chart it is seen that (0,2,8,10) & (5,7,13,15) cubes are essential prime implicant, so they are discarded from the active consideration and the resultant K-Map would be

	yz			
wx \	00	01	11	10
00	1		1	1
01		1	1	
11		1	1	
10	1			1

The remaining (2,3) and (3,7) cubes are not disjoint with the essential prime implicant. So we choose sequentially from the remaining cubes until all the minterms are covered, i.e., if we select (2,3) cube, there is no uncovered minterm. The resultant K-Map would be

	yz			
wx \	00	01	11	10
00	1		1	1
01		1	1	
11		1	1	
10	1			1

2.2.2 The Decimal Representation:

The tabulation procedure can be further simplified by adopting the decimal code for the minterms, rather than their binary representation. Two minterms can be combined only if they differ by a power of 2, i.e. only if the difference between their decimal codes is 2^i . The combined term consists of the same literals as the minterms with the exception of the variable whose weight is 2^i , which is deleted. For example, if we consider the function

$$f_1(w, x, y, z) = \sum (0, 1, 8, 9), \text{ the minterms } 1 \text{ and } 9 \text{ differ by } 2^3=8, \text{ and}$$

consequently, the variable w whose weight is 8 is deleted. This process, which is recorded by placing the weight of the redundant variable in parentheses, e.g., $1,9(8)$, is nothing but a numerical way of describing the algebraic manipulation

$$\bar{w} \cdot \bar{x} \cdot \bar{y} \cdot z + w \cdot \bar{x} \cdot \bar{y} \cdot z = \bar{x} \cdot \bar{y} \cdot z$$

Similarly, the combination of minterms 0 and 8 is written as $0,8(8)$.

The condition that the decimal codes of two combinable terms must differ by a power of 2 is necessary but not sufficient. Two terms whose codes differ by a power of 2 but which have the same index cannot be combined, since they differ by more than one variable. Similarly, if a term with a smaller index has a higher decimal value than another term whose index is higher, then the two terms cannot be combined, although they may differ by a power of 2. For example, the terms 9 and 7, whose indices are 2 and 3, respectively, cannot be combined, since they differ in the values of three variables. Except for the above phenomenon, the tabulation procedure using the decimal representation is completely analogous to that using the binary representation.

2.3 Algorithm:

1. Read the input and check whether it is the first value or not. In the case of the first value create 'head' and initiate a linked list, otherwise add the value to the linked list.
2. Continue step 1 until the end of the input.
3. Create the minterms from the values.
4. again = TRUE
5. If Again != TRUE goto step 13
6. again = FALSE, current = head
7. If current->next = NULL
 - i. current has not been combined, create a new cube and add to the new list, new_term++.

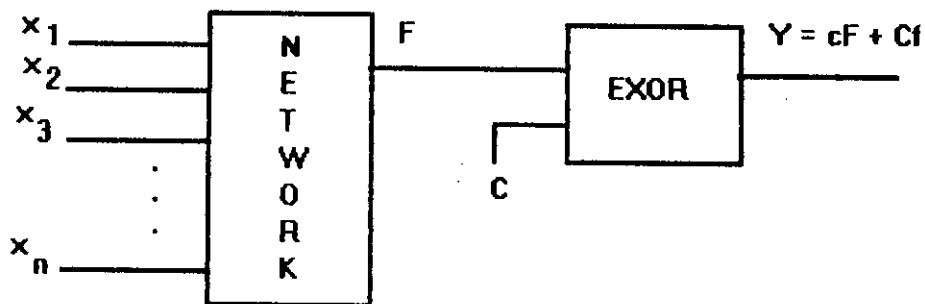
- ii. head = new_head, term_no = new_term, goto step 5.
8. link_next = current-next
9. If link_next == NULL
 - i. current has been combined, current = current-next, goto step 7.
 - ii. Create a new cube and add to the new list, new_term++, current = current-next, goto step 7.
10. Compare the two cubes, change = TRUE if the two cubes differ in one position.
11. If change == TRUE.
 - i. again = TRUE.
 - ii. combine these two cubes and create a new cube.
 - iii. new_term++.
12. link_next = link_next-next, goto step 9.
13. Check duplicity of the cubes and delete the duplicate cube.
14. Sort the prime-implicant in decreasing order of cubes.
15. Determine the essential prime implicants and output the corresponding product terms, if all the minterms are covered then stop the process.
16. Determine the disjoint prime implicant with the essential prime implicant and output the corresponding product terms. Again if all the minterms are covered then stop the process.
17. Traverse the remaining prime-implicant, sequentially and output the corresponding product term until all the minterms are covered and then stop the process.

CHAPTER 3

CLASSIFICATION OF AND-EXOR EXPRESSIONS

3.1 Introduction:

It has been conjectured that exclusive sum-of-products expressions(ESOPs) require fewer products than sum-of-products expressions(SOPs)[22]. The EXOR gate is usually more expensive to realize physically than either the AND gate or the OR gate. However, the EXOR connective is very useful since it arises naturally in the design of arithmetic circuitry. Error-detecting and error correcting circuitry also often uses EXOR gates. The parity function, the sum modulo 2 of a number of bits, is the most common type of function used in error control circuitry. An ESOP requires only n products to represent a parity function of n variables while the SOP requires 2^{n-1} [22]. If a constant 1 signal is available to use as a gate input, it is possible to form an inverter from an EXOR gate. The EXOR gate is sometimes used as a " Controlled Inverter". When the controlled signal C is 0, the network output F appears at the final output Y unchanged(fig. 3.1).



Here c is the complemented form of C
[fig. 3.1]

When the control signal is '1', the final output Y is the complement of the network output F. This connection is often provided at the outputs of a standard IC in order to increase its usefulness. Experiments using randomly generated functions show that ESOPs require, on the average, fewer products than SOPs. However, this is not always the case. There is a $2n$ variable function which requires $2^n - 1$ products in an ESOP while only n products in an SOP[22].

3.2 Several Classes of AND-EXOR Expressions:

Many researchers defined various classes of AND-EXOR expressions but the terminology is not unified[22]. In this section 7 classes of AND-EXOR expressions are introduced and the relations among them are shown. There are:

1. Positive Polarity Reed-Muller Expressions(PPRME)
2. Fixed Polarity Reed-Muller Expressions(FPRME)
3. Kronecker Expressions(KRO)
4. Pseudo Reed-Muller Expressions(PSDRME)
5. Pseudo Kronecker Expressions(PSDKRO)
6. Generalized Reed-Muller Expressions(GRME)
7. Exclusive-or sum-of-products Expressions(ESOPs)

3.3 Algebraic Identities for the EXOR-Connective:

$$1. \quad x \oplus y = x\bar{y} + \bar{x}y = (x + y)(\bar{x} + \bar{y})$$

$$2. \quad \overline{(x \oplus y)} = x \oplus \bar{y} = \bar{x} \oplus y = xy + \bar{x}\bar{y} = (\bar{x} + y)(x + \bar{y})$$

$$3 \text{ a. } \quad x \oplus x = 0 \qquad \text{b. } \quad x \oplus \bar{x} = 1$$

$$4 \text{ a. } \quad x \oplus 1 = \bar{x} \qquad \text{b. } \quad x \oplus 0 = x$$

$$5. \quad x(y \oplus z) = x.y \oplus x.z$$

$$6. \quad x + y = x \oplus y \oplus x.y = x \oplus \bar{x}.y$$

$$7. \quad x + y = x \oplus y \quad \text{if } x.y = 0$$

$$8. \quad x \oplus y = z \Rightarrow z \oplus y = x, \quad z \oplus x = y, \quad x \oplus y \oplus z = 0$$

$$9. \quad x \oplus (x + y) = \bar{x}y$$

$$10. \quad x \oplus x.y = x.\bar{y}$$

3.4 Expansion Theorem:

An arbitrary logic functions $f(x_1, x_2, \dots, x_n)$ can be represented as either

$$f = 1.f_0 \oplus x_1.f_2 \dots \dots \dots (3.1)$$

$$f = \bar{x}_1.f_2 \oplus 1.f_1 \dots \dots \dots (3.2)$$

$$f = \bar{x}_1.f_0 \oplus x_1.f_1 \dots \dots \dots (3.3)$$

$$\text{where } f_0 = f(0, x_2, x_3, \dots, x_n)$$

$$f_1 = f(1, x_2, x_3, \dots, x_n)$$

$$f_2 = f_0 \oplus f_1$$

Proof: f can be represented as $f = \bar{x}_1 f_0 + x_1 f_1$. Since $\bar{x}_1 f_0$ and $x_1 f_1$ are

mutually disjoint so $f = \bar{x}_1 f_0 + x_1 f_1$ becomes

$$f = \bar{x}_1 f_0 \oplus x_1 f_1 \dots \dots \dots (3.3)$$

replacing \bar{x}_1 by $1 \oplus x_1$ we have

$$\begin{aligned}
 f &= (1 \oplus x_1) f_0 \oplus x_1 f_1 \\
 &= 1 \cdot f_0 \oplus x_1 (f_0 \oplus f_1) \\
 &= 1 \cdot f_0 \oplus x_1 \cdot f_2 \dots \dots \dots (3.1)
 \end{aligned}$$

Replacing x_1 by $(1 \oplus \bar{x}_1)$ in equation (3.3) we have

$$\begin{aligned}
 f &= \bar{x}_1 f_0 \oplus (1 \oplus \bar{x}_1) f_1 \\
 &= \bar{x}_1 f_0 \oplus 1 \cdot f_1 \oplus \bar{x}_1 f_1 \\
 &= \bar{x}_1 \cdot f_2 \oplus 1 \cdot f_1 \dots \dots \dots (3.2)
 \end{aligned}$$

In the case of SOPs we can use only the type (3.3) expansion, often called a Shannon Expansion[22]. In the case of AND-EXOR expressions, we may use any of the three expansions. Thus the 7 classes of AND-EXOR expressions evolved.

3.4.1 Positive Polarity Reed-Muller Expressions(PPRME):

When we apply the type (3.1) expansions to all variables, we have an expression consisting of positive literals only:

$$\begin{aligned}
 a_0 \oplus a_1 x_1 \oplus \dots \oplus a_n x_n \oplus a_{12} x_1 x_2 \oplus a_{13} x_1 x_3 \oplus \dots \oplus a_{m-1} x_{n-1} x_n \oplus \\
 \dots \oplus a_{12\dots n} x_1 x_2 \dots x_n \dots \dots \dots (A)
 \end{aligned}$$

This is called a positive polarity Reed-Muller expression. Because PPRME is unique for a given function, no minimization problem exists. For example, the truth table of a function is given in the next page.

Table 3.1 Truth Table of Function M

x	y	z	M
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

$$\begin{aligned}
 M &= \bar{x} \cdot \bar{y} \cdot z \oplus \bar{x} \cdot y \cdot \bar{z} \oplus x \cdot \bar{y} \cdot \bar{z} \oplus x \cdot y \cdot \bar{z} \oplus x \cdot y \cdot z \\
 M_{x0} &= yz \oplus y\bar{z} \\
 M_{x1} &= \bar{y} \cdot \bar{z} \oplus y\bar{z} \oplus yz \\
 \therefore M_{x2} &= M_{x0} \oplus M_{x1} \\
 &= \bar{y} \cdot z \oplus \bar{y} \cdot \bar{z} \oplus y \cdot z \\
 \therefore M &= \bar{y}z \oplus y\bar{z} \oplus x(\bar{y}z \oplus \bar{y} \cdot \bar{z} \oplus yz) \\
 &= yz \oplus y\bar{z} \oplus xyz \oplus xy\bar{z} \oplus xyz \\
 \text{Again : } M_{y0} &= z \oplus xz \oplus x\bar{z} \\
 M_{y1} &= \bar{z} \oplus xz \\
 \therefore M_{y2} &= M_{y0} \oplus M_{y1} \\
 &= z \oplus x\bar{z} \oplus \bar{z} \\
 \therefore M &= z \oplus xz \oplus x\bar{z} \oplus yz \oplus xy\bar{z} \oplus y\bar{z} \\
 M_{z0} &= x \oplus xy \oplus y \\
 M_{z1} &= 1 \oplus x \oplus y \\
 \therefore M_{z2} &= 1 \oplus xy
 \end{aligned}$$

So the PPRME for the above function is

$$M = x \oplus xy \oplus y \oplus z \oplus xyz$$

The average number of product terms in the PPRMEs for the n-variable functions is 2^{n-1} .

3.4.2 Fixed Polarity Reed-Muller Expression(FPRME):

When we apply either the type (3.1) or the type (3.2) expansion to each variable, we obtain FPRME.

There are at most 2^n different FPRMEs for an n variable function. The minimization problem is to find an expression with the minimum number of products among the 2^n different FPRMEs.

For example, the exclusive-or sum of products for the given function is

$$\begin{aligned} M &= \bar{x}.\bar{y}.z \oplus \bar{x}.y.\bar{z} \oplus x.\bar{y}.\bar{z} \oplus x.y.\bar{z} \oplus x.y.z \\ M_{x_0} &= \bar{y}.z \oplus y.\bar{z} \\ M_{x_1} &= \bar{y}.\bar{z} \oplus y.\bar{z} \oplus y.z \\ M_{x_2} &= \bar{y}.z \oplus \bar{y}.\bar{z} \oplus y.z \end{aligned}$$

Applying type (3.1) we get

$$\begin{aligned} M &= \bar{y}.z \oplus y.\bar{z} \oplus x.\bar{y}.z \oplus x.\bar{y}.\bar{z} \oplus x.y.z \\ \text{Again : } M_{y_0} &= z \oplus x.z \oplus x.\bar{z} \\ M_{y_1} &= \bar{z} \oplus x.z \\ M_{y_2} &= M_{y_0} \oplus M_{y_1} \\ &= z \oplus x.\bar{z} \oplus \bar{z} \end{aligned}$$

Applying type (3.2) we get

$$\begin{aligned} M &= \bar{y}.z \oplus x.\bar{y}.\bar{z} \oplus \bar{y}.\bar{z} \oplus \bar{z} \oplus x.z \\ \text{Again : } M_{z_0} &= x.\bar{y} \oplus \bar{y} \oplus 1 \\ M_{z_1} &= \bar{y} \oplus x \\ M_{z_2} &= x.\bar{y} \oplus 1 \oplus x \end{aligned}$$

Applying type (3.1) we get the FPRME

$$M = 1 \oplus \bar{y} \oplus x.\bar{y} \oplus x.\bar{y}.z \oplus z \oplus x.z$$

3.4.3 Kronecker Expression(KRO):

When we apply either the type (3.1),(3.2) or (3.3) expansion to each variable, we obtain an expression which is more general than FPRME. This is called a Kronecker expression(KRO). There are at most 3^n different KROs for an n -variable function. To find a KRO with the minimum number of products, an extended truth table of 3^n entries and

the extended weight vector is used (See in[5]).

For example, the exclusive-or sum of products for the given function is

$$\begin{aligned}
 M &= \bar{x}. \bar{y}. z \oplus \bar{x}. y. \bar{z} \oplus x. \bar{y}. \bar{z} \oplus x. y. \bar{z} \oplus x. y. z \\
 M_{x_0} &= \bar{y}. z \oplus y. \bar{z} \\
 M_{x_1} &= \bar{y}. \bar{z} \oplus y. \bar{z} \oplus y. z \\
 M_{x_2} &= \bar{y}. z \oplus \bar{y}. \bar{z} \oplus y. z
 \end{aligned}$$

Applying type (3.1) for x variable we get

$$\begin{aligned}
 M &= \bar{y}. z \oplus y. \bar{z} \oplus x. \bar{y}. z \oplus x. \bar{y}. \bar{z} \oplus x. y. z \\
 \text{Again : } M_{y_0} &= z \oplus x. z \oplus x. \bar{z} \\
 M_{y_1} &= \bar{z} \oplus x. z \\
 M_{y_2} &= M_{y_0} \oplus M_{y_1} \\
 &= z \oplus x. \bar{z} \oplus \bar{z}
 \end{aligned}$$

Applying type (3.2) for y variable we get

$$\begin{aligned}
 M &= \bar{y}. z \oplus x. \bar{y}. \bar{z} \oplus \bar{y}. \bar{z} \oplus \bar{z} \oplus x. z \\
 \text{Again : } M_{z_0} &= x. \bar{y} \oplus \bar{y} \oplus 1 \\
 M_{z_1} &= \bar{y} \oplus x
 \end{aligned}$$

Applying type (3.3) we get the Kronecker expression for the given function

$$\begin{aligned}
 M &= \bar{z}. M_{z_0} \oplus z. M_{z_1} \\
 &= \bar{z}. x. \bar{y} \oplus \bar{y}. \bar{z} \oplus \bar{z} \oplus \bar{y}. z \oplus x. z
 \end{aligned}$$

3.4.4 Pseudo Reed-Muller Expression(PSDRME):

When we apply either the type (3.1) or the type (3.2) expansion to M we have two subfunctions. For each sub-function we can apply either type (3.1) or type (3.2) expansion. Assume that we can use different expansions for each sub-function. In this case, we have a more general expansion than a FPRME. This is called a Pseudo Reed-

Muller expression(PSDRME). In PSDRME, both true and complemented literals can appear for the same variable. There are at most 2^{2^n-1} different PSDRMEs. A minimum PSDRME can be obtained from the extended truth table. This class of expressions has not been studied[22].

For example, the exclusive-or sum of products for the given function is

$$\begin{aligned} M &= \bar{x}. \bar{y}. z \oplus \bar{x}. y. \bar{z} \oplus x. \bar{y}. \bar{z} \oplus x. y. \bar{z} \oplus x. y. z \\ M_{x0} &= \bar{y}. z \oplus y. \bar{z} \\ M_{x1} &= \bar{y}. \bar{z} \oplus y. \bar{z} \oplus y. z \\ M_{x2} &= \bar{y}. z \oplus \bar{y}. \bar{z} \oplus y. z \end{aligned}$$

Applying type (3.1) for x variable we get

$$M = (\bar{y}. z \oplus y. \bar{z}) \oplus x(\bar{y}. z \oplus \bar{y}. \bar{z} \oplus y. z)$$

Let $g = \bar{y}. z \oplus y. \bar{z}$ & $h = x. \bar{y}. z \oplus x. \bar{y}. \bar{z} \oplus x. y. z$

$$\text{Now, } g_{y0} = z, \quad g_{y1} = \bar{z}, \quad \therefore g_{y2} = z + \bar{z}$$

Applying type (3.2) in g for y variable, we have

$$\begin{aligned} g &= \bar{y}(z + \bar{z}) \oplus \bar{z} \\ \text{Let } g1 &= \bar{y}(z + \bar{z}), \quad g2 = \bar{z} \\ \therefore g1_{z0} &= \bar{y}, \quad g1_{z1} = \bar{y}, \quad g1_{z2} = 0 \\ \text{Again : } g2_{z0} &= 1, \quad g2_{z1} = 0, \quad g2_{z2} = 1 \end{aligned}$$

Applying type (3.2) in g1 & g2 for z variable we have

$$\begin{aligned} g1 &= \bar{z}. 0 \oplus \bar{y} = \bar{y}, \quad g2 = \bar{z} \\ \therefore g &= \bar{y} + \bar{z} \\ \text{Now, } h_{y0} &= x. z \oplus x. \bar{z}, \quad h_{y1} = x. z, \quad h_{y2} = x. \bar{z} \end{aligned}$$

Applying type (3.2) in function h for y variable, we have

$$\begin{aligned}
h &= \bar{y}.x.\bar{z} \oplus x.z \\
\text{Let } h1 &= \bar{y}.x.\bar{z}, \quad h2 = x.z \\
\therefore h1_{z0} &= x.\bar{y}, \quad h1_{z1} = 0, \quad h1_{z2} = x.\bar{y} \\
\text{Again : } h2_{z0} &= 0, \quad h2_{z1} = x, \quad h2_{z2} = x
\end{aligned}$$

Applying type (3.1) in h1 & type (3.2) in h2 for Z variable we have

$$\begin{aligned}
h1 &= x.\bar{y} \oplus z.x.\bar{y}, \quad h2 = \bar{z}.x \oplus x \\
\therefore h &= x.\bar{y} \oplus z.x.\bar{y} \oplus \bar{z}.x \oplus x
\end{aligned}$$

PSDRME for the given function is

$$M = \bar{y} \oplus \bar{z} \oplus x \oplus x.\bar{y} \oplus z.x.\bar{y} \oplus \bar{z}.x$$

3.4.5 Pseudo Kronecker Expression(PSDKRO):

When we apply either the type (3.1), (3.2) & (3.3) expansion to M, we have two sub-function. For each subfunction, we can apply either the type (3.1), (3.2) or (3.3) expansion. Assume that we can use different expansions for each sub-function. In this case, we have a more general expansion than a KRO. This is called a Pseudo Kronecker Expression(PSDKRO). In PSDKRO, both true and complemented literals can appear for the same variable. There are at most 3^{2^n-1} different PSDKROs. An optimization method for Pseudo Kronecker expressions of p-valued input two-valued output functions by using multiple-place decision diagrams for p=2 and p=4 is shown in[24].

For example, the exclusive-or sum of products for the given function is

$$\begin{aligned}
M &= \bar{x}.\bar{y}.z \oplus \bar{x}.y.\bar{z} \oplus x.\bar{y}.\bar{z} \oplus x.y.\bar{z} \oplus x.y.z \\
f_{x0} &= \bar{y}.z \oplus y.\bar{z} \\
f_{x1} &= \bar{y}.\bar{z} \oplus y.\bar{z} \oplus y.z
\end{aligned}$$

Applying (3.1) we have

$$\begin{aligned}
 M &= \bar{y}.z \oplus y.\bar{z} \oplus x(\bar{y}.z \oplus y.\bar{z} \oplus \bar{y}.\bar{z} \oplus y.\bar{z} \oplus y.z) \\
 &= y.z \oplus \bar{y}.\bar{z} \oplus x(\bar{y}.z \oplus y.\bar{z} \oplus \bar{y}.\bar{z} \oplus y.z) \\
 \text{Let } g &= \bar{y}.z \oplus y.\bar{z}, \quad h = x(\bar{y}.z \oplus y.\bar{z} \oplus \bar{y}.\bar{z} \oplus y.z) \\
 \text{Now, } g_{y0} &= z, \quad g_{y1} = \bar{z}, \quad \therefore g_{y2} = z \oplus \bar{z} \\
 \text{Applying (2) we have } g &= (\bar{y}.z \oplus y.\bar{z}) \oplus \bar{z} \\
 \text{Let } k &= y.z \oplus \bar{y}.\bar{z}, \quad l = z
 \end{aligned}$$

Applying (3.2) for z variable in both k & l we have

$$\begin{aligned}
 \text{Again : } k &= \bar{y}, \quad l = \bar{z}, \quad \therefore g = \bar{y} \oplus \bar{z} \\
 h_{y0} &= x.z \oplus \bar{z}.x, \quad h_{y1} = x.z, \quad h_{y2} = \bar{z}.x \\
 \text{Applying (3) we have } h &= \bar{y}(x.z \oplus \bar{z}.x) \oplus x.y.z \\
 \text{Let } i &= \bar{y}(x.z \oplus \bar{z}.x), \quad j = x.y.z
 \end{aligned}$$

Applying (3.1) in i & applying (3.2) in J we have

$$\begin{aligned}
 i &= x.\bar{y}, \quad j = \bar{z}.x.y \oplus x.y \\
 h &= x.y \oplus \bar{z}.x.y \oplus x.y
 \end{aligned}$$

Hence the PSDKRO expression for the M function is

$$M = \bar{y} \oplus \bar{z} \oplus x.\bar{y} \oplus \bar{z}.x.y \oplus x.y \quad (\text{PSDKRO})$$

3.4.6 Generalized Reed-Muller Expression(GRME):

In the expression of the type (A), if we can freely choose the polarities of the literals, then we have a more general expression than a FPRME. This is called a Generalized Reed-Muller expression(GRME). It is also called inconsistent canonical form [5] or a canonical restricted mixed polarity form. There are at most $2^{n2^{n-1}}$ different GRMEs.

3.4.7 Exclusive-or Sum-of-Products Expressions (ESOP):

Arbitrary product terms combined by EXORs are called an Exclusive-or Sum-of-Products Expression(ESOP). The ESOP is the most general AND-EXOR expression. There are at most 3^t different ESOPs, where t is the number of the products. No efficient minimization method is known and iterative improvement methods are used to obtain near minimal solution[5]. EXMIN2, a heuristic simplification Algorithm[5], would be used to minimize such expression in a later chapter.

3.5 Relations among the classes:

Suppose that PPRME, FPRME, PSDRME, KRO, PSDKRO, GRME and ESOP denote the set of expressions. Then the following relations hold:

- i. PPRME \subset FPRME(From Art. 1 & Art. 2)
- ii. FPRME \subset PSDRME(From Art. 2 & Art. 4)
- iii. PSDRME \subset GRME(From Art. 4 & Art. 6)
- iv. FPRME \subset KRO(From Art. 2 & Art. 3)
- v. KRO \subset PSDKRO(From Art. 3 & Art. 5)
- vi. PSDRME \subset PSDKRO(From Art. 4 & Art. 5)

CHAPTER 4

LOGIC SYNTHESIS WITH EXOR GATES

4.1 Introduction:

Increasing complexity of LSIs has made human design of bug-free logic circuits very difficult. Thus various automatic logic synthesis tools have become indispensable in LSI design. Most of the logic synthesis tools use the design theory for the circuits consisting of AND, OR and NOT gates. As for control circuits these tools produce good circuits comparable to the human design. However, they are not so good at the design of arithmetic circuits, error correcting circuits and circuits for telecommunication: such circuits can be simplified when EXOR gates are effectively used. These circuits can be realized with many fewer gates if EXOR gates are available as well as AND and OR gates and can be derived from AND-EXOR two level circuits (AND-EXORs). Logic design using exclusive-OR gates offers two advantages when compared to conventional Boolean realizations using AND/OR/NAND/NOR gates. Firstly, in certain cases more economic realizations in terms of the number of gates and/or interconnections may be obtained. Secondly, the testability of circuits is significantly improved[17]. Therefore, in order to develop a logic synthesis tool for such circuits, a design theory utilizing EXOR gates is very important.

4.2 Design Method of AND-EXOR Circuits:

In this section, we will review the design method for AND-ORs and AND-EXORs. The symbol \vee denotes the inclusive-OR operation, while \oplus denotes the exclusive-OR operation.

Definition-1:

Products combined by OR is a sum-of-products expression(SOP). Products combined by EXOR is an exclusive-or sum-of-products expression(ESOP). An SOP(ESOP) for a function f with the minimum number of products is called a minimum SOP(minimum ESOP) and denoted by MSOP(MESOP).

Fig. 4.1 and fig. 4.2 show an MSOP and an MESOP for a 4-variable function, respectively. In the minimization of SOPs, each

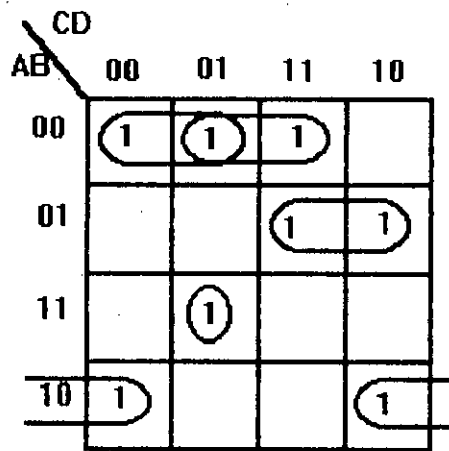


fig. 4.1

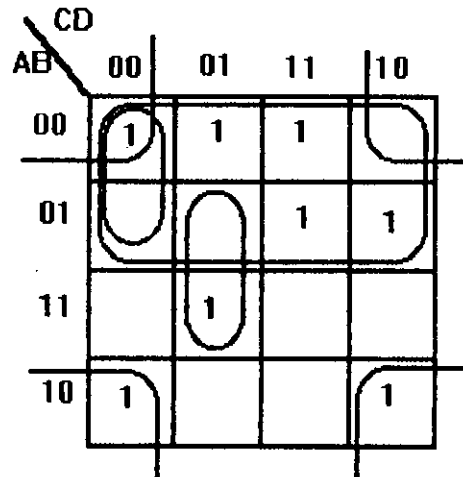


fig. 4.2

minterm of the function must be covered by loop(s) at least once(fig. 4.1). However, in the minimization of ESOPs, each minterm of the function must be covered by loop(s) by odd times(fig. 4.2). This is due to the fact that $1 \vee 1 = 1$ in SOPs, but $1 \oplus 1 = 0$ in ESOPs. Fig. 4.2 requires fewer loops than fig. 4.1 and the loops in fig. 4.2 are larger than ones in fig. 4.1. This shows the two-level realization based on the ESOP requires fewer gates and connections than one based on the SOP.

4.3 Simplification of AND-EXOR Expressions:

In the minimization of SOPs, we must cover each minterm of the function by loop(s) at least once. However, in the minimization of ESOPs, we must cover each minterm of the function by the loops in odd times. We can include the terms of zeros in the truth table

by loops in even times. So the minimization of ESOPs is much more difficult than that of SOPs. In the minimization of SOPs, concepts such as prime implicants and essential prime implicants are very useful. However, in the minimization of ESOPs, we cannot use such concepts. This is the reason why exact minimization programs can treat the functions with less than 5 or 6 inputs(See in [16],[10]).

In order to treat functions with more variables, we have to abandon the absolute minimalities and must resort to near minimal solutions. Various heuristic tools have been developed[5].

A heuristic simplification algorithm EXMIN2 for ESOPs with multi-valued inputs, reduces the number of products in ESOPs as the first objective, and then reduces the number of literals as the second objectives[23]. In this section the outline of EXMIN2 for two valued input function would be elaborately described and the software would be developed.

Definition 2:

An SOP is called a Disjoint SOP(DSOP) if all the products are mutually disjoint.

For example:

The truth table of an arbitrary function is shown below

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

K-Map of the above truth table

		BC			
A		00	01	11	10
0			1		1
1		1			1

if we circle each minterm individually then the K-Map would be

		BC			
A		00	01	11	10
0			(1)		(1)
1		(1)			(1)

So the minterms of the truth table are mutually disjoint.

		BC			
A		00	01	11	10
0			(1)		(1)
1		(1)			(1)

fig(a)

		BC			
A		00	01	11	10
0			(1)		(1)
1		(1)			(1)

fig(b)

fig.(a) is mutually disjoint but fig.(b) is not mutually disjoint.

Lemma-1:

In a DSOP, the OR operators can be replaced by the EXOR operators without changing function.

Proof:

EXOR operation is defined as

$$x_1 \oplus x_2 = x_1 \bar{x}_2 \vee \bar{x}_1 x_2$$

and the inclusive OR is represented by EXOR as

$$x_1 \vee x_2 = x_1 \oplus x_2 \oplus x_1 x_2$$

Thus if $x_1 x_2 = 0$, then $x_1 \vee x_2 = x_1 \oplus x_2$

In general if $x_i x_j = 0$ for $(1 \leq i \leq j \leq k)$, then

$$x_1 \vee x_2 \vee \dots \vee x_k = x_1 \oplus x_2 \oplus \dots \oplus x_k = Y$$

Again if $x_i x_j = 0$ for $(1 \leq i \leq j \leq k+1)$, then $Y \vee x_{k+1} = 0$

Thus, $Y \vee x_{k+1} = Y \oplus x_{k+1}$

Hence we have

$$x_1 \vee x_2 \vee \dots \vee x_k \vee x_{k+1} = x_1 \oplus x_2 \oplus \dots \oplus x_{k+1}$$

The minterms of the given function are mutually disjoint. So the initial solutions of the algorithm are DSOPs derived from the truth table which are the minterms of the function. Then according to Lemma-1, we can replace the ORs with the EXORs without changing the function.

Definition-3:

Let x be a variable that takes one of the values in $P = \{0, 1, \dots, p-1\}$. For any subset $S \subseteq P$, x^S is a literal representing the function that

$$x^s = 1 \quad \text{if } x \in S$$

$$0 \quad \text{if } x \notin S$$

If $P = \{0, 1\}$, then the literals are $x^{(0,1)}, x^{(0)}, x^{(1)}$ and x^\emptyset which may be denoted by $1, \bar{x}, x$ and \emptyset respectively.

Volume of Cubes:

Definition: Let $|S|$ denote the number of elements in S . The volume of a product $x_1^{s_1}, x_2^{s_2}, \dots, x_n^{s_n}$ is $|S_1| \cdot |S_2| \cdot \dots \cdot |S_n|$. The volume of an ESOP is the sum of volumes of products in the ESOP.

In the case of two-valued input single-output functions, the greater the volume, the fewer are the literals in the ESOP's. So the increase of the volume is desirable for the reduction of the circuit cost.

Some useful algebraic identities for \oplus -connectives are:

a) $x \oplus y = xy + \bar{x}y = (x + y)(\bar{x} + \bar{y})$

b) $\overline{x \oplus y} = x \oplus \bar{y} = \bar{x} \oplus y = x \cdot y + \bar{x} \cdot \bar{y} = (\bar{x} + y)(x + \bar{y})$

c) i) $x \oplus x = 0$ ii) $x \oplus \bar{x} = 1$

d) i) $x \oplus 1 = \bar{x}$ ii) $x \oplus 0 = x$

e) $x(y \oplus z) = xy \oplus xz$

f) $x + y = x \oplus y \oplus xy = x \oplus \bar{x}y$

g) $x + y = x \oplus y$ if $xy = 0$

h) $x \oplus y = z \Rightarrow z \oplus y = x, z \oplus x = y, x \oplus y \oplus z = 0$

$$i) x \oplus (x + y) = \bar{x}y$$

$$j) x \oplus xy = x\bar{y}$$

The algorithm used in this section uses the following properties:

Associative Law :

Definition: A binary operator * on a Set S is said to be associative whenever:

$$(x * y) * z = x *(y * z) \text{ for all } x, y, z \in S$$

For EXOR operation this would be

$$(x \oplus y) \oplus z = x \oplus (y \oplus z)$$

Commutative Law :

Definition: A binary operator * on a set S is said to be commutative whenever :

$$x * y = y * x \text{ for all } x, y \in S$$

For EXOR operation this would be

$$x \oplus y = y \oplus x$$

Distributive Law :

Definition: If * and & are two binary operators on a set S, * is said to be distributive over & whenever

$$x *(y \& z) = (x * y)\&(x * z)$$

For EXOR operation if '.', i.e., AND is said to be distributive over \oplus whenever

$$(x \oplus y).z = x.z \oplus y.z$$

In addition to these rules, Algorithm used in this chapter uses the following rules to replace a pair of products with another one.

1) X-MERGE:

$$\text{a) } x \oplus x = 0 \quad \text{b) } x \oplus \bar{x} = 1 \quad \text{c) } x \oplus 1 = \bar{x} \quad \text{d) } \bar{x} \oplus 1 = x$$

These rules do not increase the volume but reduce the number of the products

Proof: $x \oplus x = x \cdot \bar{x} \vee \bar{x} \cdot x = 0$

$$x \oplus \bar{x} = x \cdot x \vee \bar{x} \cdot \bar{x} = x \vee \bar{x} = 1$$

$$x \oplus 1 = x \cdot 0 \vee \bar{x} \cdot 1 = \bar{x}$$

$$\bar{x} \oplus 1 = \bar{x} \cdot 0 \vee x \cdot 1 = x$$

2) RESHAPE :

$$x \cdot y \oplus \bar{y} = \bar{x} \cdot \bar{y} \oplus x$$

This rule does not change the volume.

Proof: From (h) we get

$$A \oplus B = C \oplus D \quad \text{if } A \oplus B \oplus C \oplus D = 0$$

So let $A = (x \cdot y \oplus \bar{y}) \oplus (\bar{x} \cdot \bar{y} \oplus x)$

$$= (x \cdot y \oplus \bar{x} \cdot \bar{y}) \oplus (x \oplus \bar{y})$$

$$= (x \oplus \bar{y}) \oplus (x \oplus \bar{y}) \quad \text{..... [Using X-EXPAND1]}$$

$$= (x \oplus x) \oplus (\bar{y} \oplus \bar{y})$$

$$= 0$$

$$\text{i.e. } x \cdot y \oplus \bar{y} = \bar{x} \cdot \bar{y} \oplus x$$

3) DUAL-COMPLEMENT:

$$x \oplus y = \bar{x} \oplus \bar{y}$$

DUAL-COMPLEMENT does not change the volume for two-valued input functions but may decrease the volume for multi-valued-input functions (See in [23]).

Proof: Let $A = (x \oplus y) \oplus (\bar{x} \oplus \bar{y})$

$$= (x \oplus \bar{x}) \oplus (y \oplus \bar{y})$$

$$= 1 \oplus 1$$

$$= 0$$

$$\text{i.e. } x \oplus y = \bar{x} \oplus \bar{y}$$

4) X-EXPAND1:

$$x \cdot \bar{y} \oplus \bar{x} \cdot y = x \oplus y$$

X-EXPAND1 increases the volume i.e reduces the number of the literals in the ESOPs and hence number of connections are reduced.

Proof: Let $A = (x \cdot \bar{y} \oplus \bar{x} \cdot y) \oplus (x \oplus y)$

$$= (x \oplus x \cdot \bar{y}) \oplus (y \oplus \bar{x} \cdot y)$$

$$= x(1 \oplus \bar{y}) \oplus y(1 \oplus \bar{x})$$

$$=x.y \oplus y.x \quad \dots\dots \text{[Using X-MERGE]}$$

$$= 0$$

$$\text{i.e., } x.\bar{y} \oplus \bar{x}.y = x \oplus y$$

5) X-EXPAND2:

$$x.y \oplus \bar{y} = 1 \oplus \bar{x}.y$$

X-EXPAND2 also increases the volume i.e. reduces the number of the literals in the ESOPs.

$$\text{Proof: Let } A=(x.y \oplus \bar{y}) \oplus (1 \oplus \bar{x}.y)$$

$$=(\bar{y} \oplus 1) \oplus (x.y \oplus \bar{x}.y)$$

$$=y \oplus y(x \oplus \bar{x}) \quad \dots\dots\dots \text{[Using X-MERGE]}$$

$$=y \oplus y$$

$$= 0$$

$$\text{i.e., } x.y \oplus \bar{y} = 1 \oplus \bar{x}.y$$

6) X-REDUCE1:

$$x \oplus y = x.\bar{y} \oplus \bar{x}.y$$

X-REDUCE1 is the reverse operation of X-EXPAND1. So X-REDUCE1 decreases the volume of the cubes, i.e., increases the number of literals.

7) X-REDUCE2:

$$1 \oplus \bar{x} \cdot y = x \cdot y \oplus \bar{y}$$

X-REDUCE2 is also the reverse operation of X-EXPAND2 and decreases the volume of the cubes i.e. increases the number of literals.

8) SPLIT :

$$1 = x \oplus \bar{x}$$

SPLIT increases the number of products as well as number of literals in the product. Among the above rules, X-MERGE & X-EXPANDs will simplify the ESOPs. However these rules are not sufficient to produce the minimum solutions [23]. Other rules will prevent the algorithm from falling into local minima[5].

4.4 Algorithm:

- a) Formation of exclusive-or-sum of products of minterm from the truth table.
- b) For each pair of products, do X-MERGE.
- c) For each pair of products, do RESHAPE, DUAL-COMPLEMENT, X-EXPAND2 and X-EXPAND1. For the products modified by these rules, do X-MERGE.
- d) If X-EXPAND1 or X-EXPAND2 is applied in (c), then do (c) again.
- e) For each pair of products, do X-MERGE, again.
- f) Apply X-REDUCE1 and X-REDUCE2.
- g) Do (b)-(e) again.
- h) If the number of products is reduced in (g), then go to (c).
- i) Increase the number of products by SPLIT: for each variable x_i , expand the ESOP F into $\bar{x}_i \cdot F_i \oplus x_i \cdot F_i$. Find a variable x_i that increases the minimum number of products in $\bar{x}_i \cdot F_i \oplus x_i \cdot F_i$. Simplify each subfunction independently by (b)-(h).

Then, simplify the total function again. Apply this algorithm as long as the reduction of the number of products is possible.

Example-1: Consider the function Y in the following truth table.

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Step (a):

$$Y = \bar{A} \bar{B} \bar{C} \bar{D} \oplus \bar{A} \bar{B} \bar{C} D \oplus \bar{A} \bar{B} C D \oplus \bar{A} B C \bar{D} \oplus \bar{A} B C D \oplus A \bar{B} \bar{C} \bar{D} \oplus A \bar{B} C \bar{D} \oplus A B \bar{C} D$$

K-Map:

		CD			
		00	01	11	10
AB	00	1	1	1	
	01			1	1
	11		1		
	10	1			1

Step b:

$$\begin{aligned}
 Y &= \bar{A} \bar{B} \bar{C} D \oplus \bar{D} \oplus \bar{A} C D \bar{B} \oplus B \oplus \bar{A} B C \bar{D} \oplus A \bar{B} \bar{D} C \oplus \bar{C} \\
 &= \bar{A} \bar{B} \bar{C} \oplus \bar{A} C D \oplus \bar{A} B C \bar{D} \oplus A \bar{B} \bar{D} \oplus A B \bar{C} D
 \end{aligned}$$

K-Map:

	CD				
AB	00	01	11	10	
00	1	1	1		(1)
01			1	1	(2)
11		1			
10	1			1	

Step c:

Applying RESHAPE in (1) and (2) we have

$$\begin{aligned}
 Y &= \bar{A} \bar{B} \bar{C} \oplus \bar{A} C D \oplus B \bar{D} \oplus A \bar{B} \bar{D} \oplus A B \bar{C} D \\
 &= \bar{A} \bar{B} \bar{C} \oplus \bar{A} C (\bar{B} D \oplus B) \oplus A \bar{B} \bar{D} \oplus A B \bar{C} D \\
 &= \bar{A} \bar{B} \bar{C} \oplus \bar{A} B C \oplus \bar{A} \bar{B} C D \oplus A \bar{B} \bar{D} \oplus A B \bar{C} D
 \end{aligned}$$

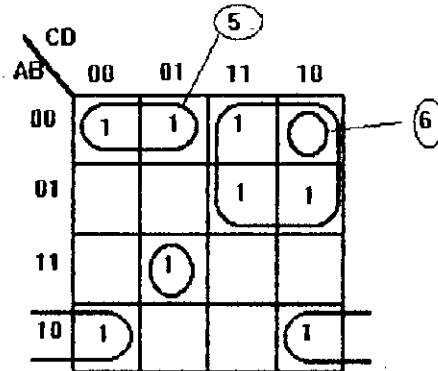
K-Map:

	CD				
AB	00	01	11	10	
00	1	1	1		(3)
01			1	1	(4)
11		1			
10	1			1	

we can not apply DUAL-COMPLEMENT but applying X-EXPAND2 in (3) and (4)

$$\begin{aligned}
 Y &= \bar{A} \bar{B} \bar{C} \oplus \bar{A} C \bar{B} \bar{D} \oplus B \oplus A \bar{B} \bar{D} \oplus A B \bar{C} D \\
 &= \bar{A} \bar{B} \bar{C} \oplus \bar{A} C (\bar{B} \bar{D} \oplus 1) \oplus A \bar{B} \bar{D} \oplus A B \bar{C} D \\
 &= \bar{A} \bar{B} \bar{C} \oplus \bar{A} \bar{B} C \bar{D} \oplus \bar{A} C \oplus A \bar{B} \bar{D} \oplus A B \bar{C} D
 \end{aligned}$$

K-Map:



we cannot apply X-EXPAND1 and we cannot apply X-MERGE also.

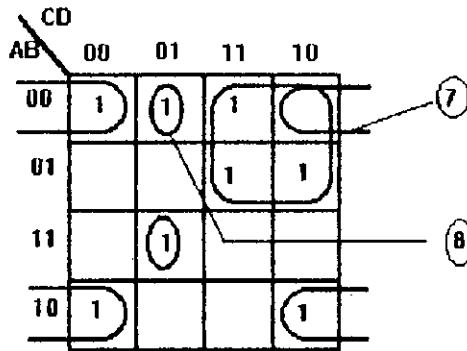
Step d:

X-EXPAND2 is applied in step c, So do step c again.

Applying RESHAPE in (5) and (6)

$$\begin{aligned}
 Y &= \bar{A} \bar{B} (\bar{C} \oplus C \bar{D}) \oplus \bar{A} C \oplus A \bar{B} \bar{D} \oplus A B \bar{C} D \\
 &= \bar{A} \bar{B} (\bar{D} \oplus \bar{C} D) \oplus \bar{A} C \oplus A \bar{B} \bar{D} \oplus A B \bar{C} D \\
 &= \bar{A} \bar{B} \bar{D} \oplus \bar{A} \bar{B} \bar{C} D \oplus \bar{A} C \oplus A \bar{B} \bar{D} \oplus A B \bar{C} D
 \end{aligned}$$

K-Map:



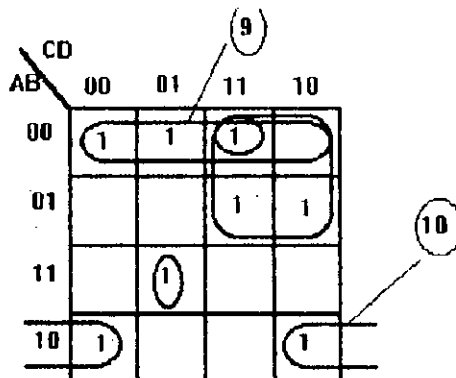
we cannot apply DUAL-COMPLEMENT but applying X-EXPAND2 in (7) and (8)

$$Y = \bar{A} \bar{B} \bar{D} \oplus \bar{C} D \oplus \bar{A} C \oplus A \bar{B} \bar{D} \oplus A B \bar{C} D$$

$$= \bar{A} \bar{B} (C D \oplus 1) \oplus \bar{A} C \oplus A \bar{B} \bar{D} \oplus A B \bar{C} D$$

$$= \bar{A} \bar{B} C D \oplus \bar{A} \bar{B} \oplus \bar{A} C \oplus A \bar{B} \bar{D} \oplus A B \bar{C} D$$

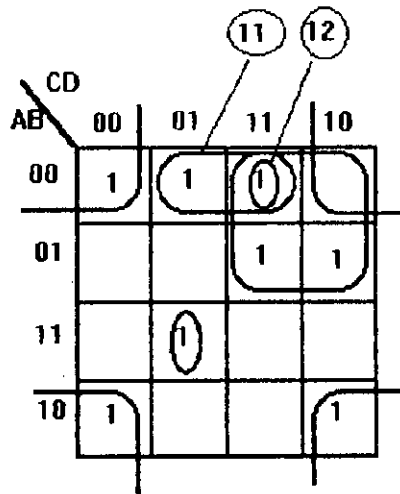
K-Map:



we cannot apply X-EXPAND1, X-MERGE. Now we can apply RESHAPE between (9) and (10) and we have

$$Y = \bar{A} \bar{B} C D \oplus \bar{B} \bar{D} \oplus \bar{A} C \oplus \bar{A} \bar{B} D \oplus A B \bar{C} D$$

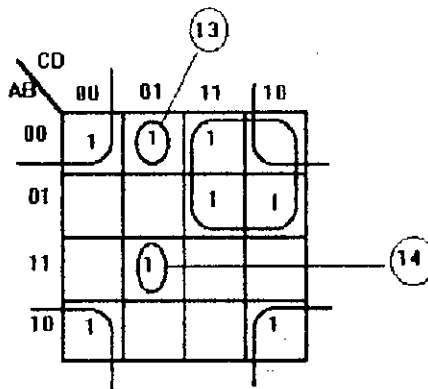
K-Map:



We cannot apply DUAL-COMPLEMENT, X-EXPAND2, X-EXPAND1. Now applying X-MERGE between (11) and (12) we get

$$Y = \bar{A} \bar{B} \bar{C} D \oplus \bar{B} \bar{D} \oplus \bar{A} C \oplus A B \bar{C} D$$

K-Map:



Step e:

We cannot apply X-MERGE

Step f:

We cannot apply X-REDUCE1 and X-REDUCE2

Step g:

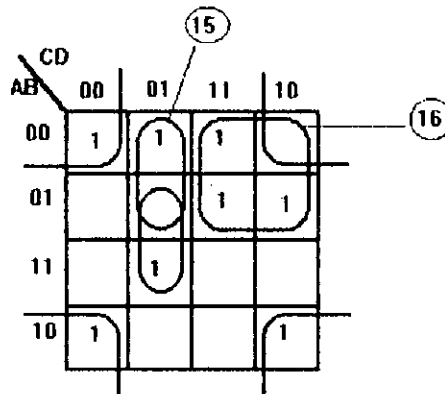
Do (b)-(e). again.

We cannot apply X-MERGE, RESHAPE, DUAL-COMPLEMENT, X-EXPAND2 but we

can apply X-EXPAND1 in (13) and (14) and we have

$$Y = \bar{A} \bar{C} D \oplus \bar{B} \bar{D} \oplus \bar{A} C \oplus B \bar{C} L$$

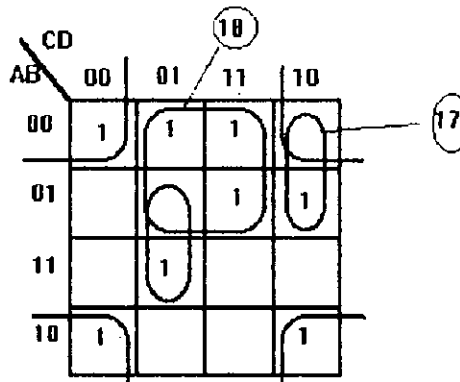
K-Map:



We cannot apply X-MERGE but we can apply RESHAPE in (15) and (16) and we have

$$Y = \bar{A} C \bar{D} \oplus \bar{B} \bar{D} \oplus \bar{A} D \oplus B \bar{C} L$$

K-Map:



We cannot apply DUAL-COMPLEMENT but we can apply X-EXPAND2 in (17) and (18) and we have

$$Y = \bar{A} \oplus \bar{B} \bar{D} \oplus \bar{A} \bar{C} \bar{D} \oplus B \bar{C} L$$

88063

K-Map:

	CD			
AB	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11		1		
10	1			1

We cannot apply X-MERGE, RESHAPE, DUAL-COMPLEMENT, X-EXPAND2, X-EXPAND1 and X-MERGE

Step h:

We cannot reduce the number of products which is 4 as before.

Step i:

Now we have to increase the number of products by SPLIT. We can increase the above expression in the following ways

i) $Y = \bar{A} \cdot Y_i \oplus A \cdot Y_i$

ii) $Y = \bar{B} \cdot Y_i \oplus B \cdot Y_i$

iii) $Y = \bar{C} \cdot Y_i \oplus C \cdot Y_i$

iv) $Y = \bar{D} \cdot Y_i \oplus D \cdot Y_i$

Among them we have to choose one which increases the minimum number of products.

Now,

$$i) Y = \bar{A} \oplus \bar{B} \cdot \bar{D}(A \oplus \bar{A}) \oplus \bar{A} \cdot \bar{C} \cdot \bar{D} \oplus B \cdot \bar{C} \cdot D(A \oplus \bar{A})$$

$$= (\bar{A} \oplus \bar{A} \cdot \bar{B} \cdot \bar{D} \oplus \bar{A} \cdot \bar{C} \cdot \bar{D} \oplus \bar{A} \cdot B \cdot \bar{C} \cdot D) \oplus (A \cdot \bar{B} \cdot \bar{D} \oplus A \cdot B \cdot \bar{C} \cdot D)$$

So number of products are 6.

$$ii) Y = \bar{A}(B \oplus \bar{B}) \oplus \bar{B} \cdot \bar{D} \oplus \bar{A} \cdot \bar{C} \cdot \bar{D}(B \oplus \bar{B}) \oplus B \cdot \bar{C} \cdot D$$

$$= (\bar{A} \cdot \bar{B} \oplus \bar{B} \cdot \bar{D} \oplus \bar{A} \cdot \bar{C} \cdot \bar{D} \cdot \bar{B}) \oplus (\bar{A} \cdot B \oplus \bar{A} \cdot \bar{C} \cdot \bar{D} \cdot B \oplus B \cdot \bar{C} \cdot D)$$

So number of products are 6.

$$iii) Y = \bar{A}(C \oplus \bar{C}) \oplus \bar{B} \cdot \bar{D}(C \oplus \bar{C}) \oplus \bar{A} \cdot \bar{C} \cdot \bar{D} \oplus B \cdot \bar{C} \cdot D$$

$$= (\bar{A} \cdot \bar{C} \oplus \bar{B} \cdot \bar{D} \cdot \bar{C} \oplus \bar{A} \cdot \bar{C} \cdot \bar{D} \oplus B \cdot \bar{C} \cdot D) \oplus (\bar{A} \cdot C \oplus \bar{B} \cdot \bar{D} \cdot C)$$

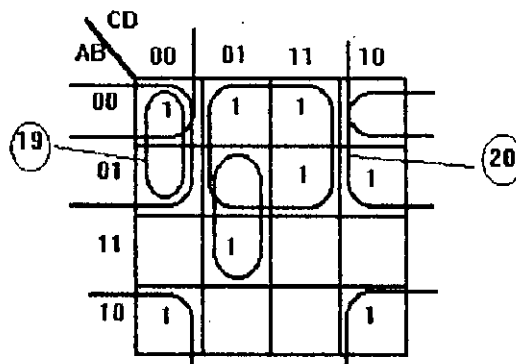
So number of products are 6.

$$iv) Y = \bar{A}(D \oplus \bar{D}) \oplus \bar{B} \cdot \bar{D} \oplus \bar{A} \cdot \bar{C} \cdot \bar{D} \oplus B \cdot \bar{C} \cdot D$$

$$= (\bar{A} \cdot \bar{D} \oplus \bar{B} \cdot \bar{D} \oplus \bar{A} \cdot \bar{C} \cdot \bar{D}) \oplus (\bar{A} \cdot D \oplus B \cdot \bar{C} \cdot D)$$

So the number of products is 5, i.e., variable D increases the number of products by the smallest amount. As a result SPLIT would use variable D.

K-Map:



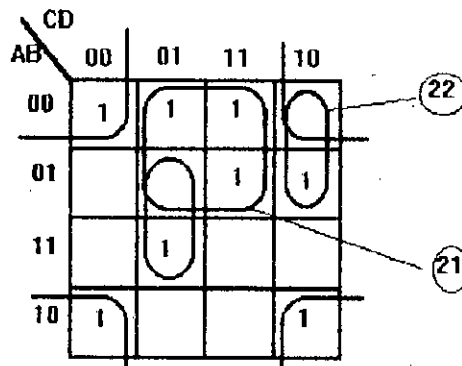
Now we would apply Step (b)-Step (h) to two subfunction independently

$$\text{Let } Y_1 = \bar{D}(\bar{A} \oplus \bar{B} \oplus \bar{A} \bar{C})$$

Applying X-MERGE between (19) and (20), we get

$$Y_1 = \bar{D}(\bar{A} C \oplus \bar{B})$$

K-Map:



We cannot apply RESHAPE, DUAL-COMPLEMENT, X-EXPAND2, X-EXPAND1. We cannot also apply X-MERGE, X-REDUCE1, X-REDUCE2. So after step (b)-step(h) we have

$$Y_1 = \bar{D}(\bar{A} C \oplus \bar{B})$$

$$\text{Now let } Y_2 = \bar{A} D \oplus B D \bar{C}$$

We cannot apply any rules of step(b)-step(h), so the simplified second subfunction is

$$Y_2 = \bar{A} D \oplus B D \bar{C}$$

The total function is $Y = Y_1 \oplus Y_2$

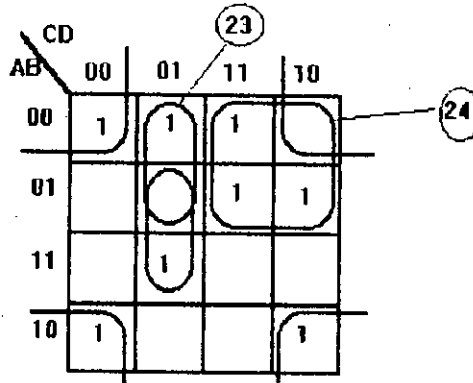
$$\text{i.e. } Y = \bar{D}(\bar{A} C \oplus \bar{B}) \oplus \bar{A} D \oplus B D \bar{C}$$

Now we would apply step(b)-step(h) in the total function

We cannot apply step(b) of the algorithm. Applying RESHAPE between (21) and (22) of step(c) we get

$$Y = \bar{A} \bar{C} D \oplus \bar{B} \bar{D} \oplus \bar{A} C \oplus B \bar{C} L$$

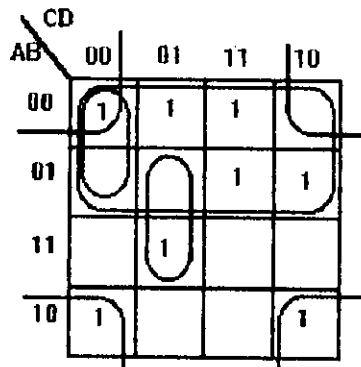
K-Map:



Applying X-EXPAND2 of step(c) between volume(23) and volume(24) we get

$$Y = \bar{A} \oplus \bar{B} \bar{D} \oplus \bar{A} \bar{C} \bar{D} \oplus B \bar{C} L$$

K-Map:



which is the same as before applying the step(i) i.e. SPLIT rules.

So this is the minimal ESOP for the given function.

CHAPTER 5

MINIMIZATION USING REED-MULLER CANONIC EXPANSION

5.1 Introduction:

The Reed-Muller canonical expansion is the basis for a family of error-correcting codes called Reed-Muller codes[6]. Switching function realized by exclusive-OR gates posses the feature of being easily tested[1]. Using exclusive-OR gates in logic design has some advantages over the conventional realizations using AND/OR/NAND/NOR gates, because many useful functions have a high content of exclusive-OR. The conjecture that switching functions represented by modulo 2 sums of products on the average require fewer products is studied [25]. The EXOR circuit is easier to test and may require fewer gates and interconnections and also have a wide application for error detection and correction in digital communication systems. For the case of EXOR, sum of products of the Reed-Muller expansion may be put to an efficient use. The Reed-Muller canonic(RMC) forms are based on Boolean EXOR and AND logic. However, the fixed/positive(true) 'polarity' of the variables in the Reed-Muller expansion proves to be a major confinement. Actually the minimum form may require different polarities of the variables in its various terms. It is generally accepted that functions which do not produce efficient solutions using other minimization techniques tend to have efficient solutions using Reed-Muller techniques. Thus minimization of Reed-Muller polynomials has been an area of recent research interest[1].

5.2 Historical Perspective for Minimization of Reed-Muller Canonic Expansions:

Before studying the minimization of Boolean functions using Reed-Muller canonic expansions, it may be helpful to put things in historical perspective. A method for realizing arbitrary combinational switching functions resulting in easily testable network was proposed by Reedy(see in [19]) in 1972. In the same year Sowmitri Swamy published a paper in which he first proposed the idea that all canonical forms are sequentially produced in the natural order, by step by step shifting upwards (See in [18]). But in 1979 Kewal K. Saluja and E.H. Ong published a paper in which it is shown that Swamy's previous approach to generate generalized Reed-Muller canonic expansion is in error. A different algorithm is presented which uses a single Boolean matrix and successive modifications in function vector to generate all the solutions sequentially (See in [20]). In 1982 Wu Chen and Hurst have suggested a map method for a generalized Reed-Muller (GRM) expansion in which the variables may be true or complemented (See in [32]). Mapping of the RM coefficients and performing the map operations (foldings) in order to get the various sets of GRM coefficient becomes troublesome if the number n of variables exceeds four. To overcome these difficulties Ph.W. Besslich proposed a fast in-place transform method for the generation of all possible sets of GRM coefficients from which the optimum may then be selected(See in [17]). In 1983, Marian Mlynarovic published (See in [3]) a method whose main idea is the same as that of Mukhopadhyay, A. and Schmitz, G. (See in [15]). This yields an algorithm that is easy for software implementation.

A graphical method for minimizing RM polynomials in mixed polarity has been introduced by A. Tran in 1987. In 1989 he implemented the above method using tristate maps (See in [26]). However, the method depends on the somewhat intuitive human ability to recognize the map pattern. In addition, it can be applied only to functions of six or fewer variables. To alleviate these limitations, A. Tran and E. Lee generalized the concept of tri-state map to functions of more than six variables and a tabular method of minimization is developed based on this algorithm. The method adopts the bottom-up

approach and will be called the 'Composition' method (See in [1]). At the same time, A. Tran and J. Wang published (See in [28]) a new method known as 'Decomposition' method which uses the top down approach. Both methods can be implemented on computers.

5.3 Reed-Muller Expansion:

The positive canonic RM expansion from the truth table can be written

$$f(x_{n-1}, x_{n-2}, \dots, x_2, x_1, x_0) = b_0 \oplus b_1 x_0 \oplus b_2 x_1 \oplus b_3 x_0 x_1 \oplus b_4 x_2 \oplus \dots \oplus b_{2^n-1} x_0 x_1 \dots x_{n-1}$$

where PPRM coefficients b_i are allocated in the order of the decimal equivalent of the x_i product terms, x_0 being the least significant bit and

$$b_i = 0 \text{ or } 1$$

$$i=0,1,2,\dots,2^n-1$$

Hence they indicate which products of the variables are present in the expansion.

If each of the n variables can appear in its true or complemented form, but not both, then fixed polarity Reed-Muller expansion results. The fixed polarity Reed-Muller expression can be written as

$$f(x_{n-1}, x_{n-2}, \dots, x_1, x_0) = b_0 \oplus b_1 x_0^* \oplus b_2 x_1^* \oplus b_3 x_0^* x_1^* \oplus \dots \oplus b_{2^n-1} x_0^* x_1^* \dots x_{n-1}^*$$

where x^* represents x or \bar{x}

The different RM expansions are identified by a polarity number (See in [11]). To calculate the polarity of any function, each variable is used in true or complemented form respectively. The polarity is the decimal equivalent of the resulting binary number. Accordingly, the positive polarity expansion has zero polarity. For example

$F(x_2, x_1, x_0)$ has polarity 0

$F(x_2, x_1, \bar{x}_0)$ has polarity 1

$F(x_2, \bar{x}_1, x_0)$ has polarity 2

$F(\bar{x}_2, \bar{x}_1, \bar{x}_0)$ has polarity 7

Any fixed polarity can be obtained by substituting variable x by $(1 \oplus \bar{x})$ and simplifying it or directly from the truth table using the three expansion theorems described in the previous chapter. For Pseudo Reed-Muller expression (PSDRME) and GRME more expansion becomes possible and the designer is left to wonder which of the many possible expansions is the most economical. The problem becomes more intractable if multi-outputs are considered. The criterion for minimization is the reduction of the number of products in the expansion and also the reduction of number of literals in the expansion. Since exhaustive search is not practicable except for very small number of variables the quest for an easy technique for minimization or conversion between Boolean and Reed-Muller expansion will continue to keep researchers busy for sometime to come.

5.4 Techniques for Minimization of RMC Expansions:

These are different fundamental techniques for minimization of a RM expression. In this section some of these techniques are illustrated.

5.4.1 Map Simplification of Positive Polarity Expansion:

The positive polarity Reed-Muller expansion for n variables can be written as follows

$$f(x_{n-1}, x_{n-2}, \dots, x_2, x_1, x_0) = b_0 x_{n-1}^0 x_{n-2}^0 \dots x_0^0 \oplus b_1 x_{n-1}^0 x_{n-2}^0 \dots x_0^1 \oplus b_2 x_{n-1}^0 x_{n-2}^0 \dots x_1^1 x_0^0 \oplus \dots \oplus b_{2^{n-1}} x_0^1 x_1^1 \dots x_{n-1}^1$$

Now subscripts of the b-coefficients correspond to the decimal equivalent of the superscripts of the variables. For a four-variable function, we have

$$f(x_3, x_2, x_1, x_0) = b_0 x_3^0 x_2^0 x_1^0 x_0^0 \oplus b_1 x_3^0 x_2^0 x_1^0 x_0^1 \oplus b_2 x_3^0 x_2^0 x_1^1 x_0^0 \oplus \dots \oplus b_{15} x_3^1 x_2^1 x_1^1 x_0^1$$

A map can be drawn for the b-coefficients as follows:

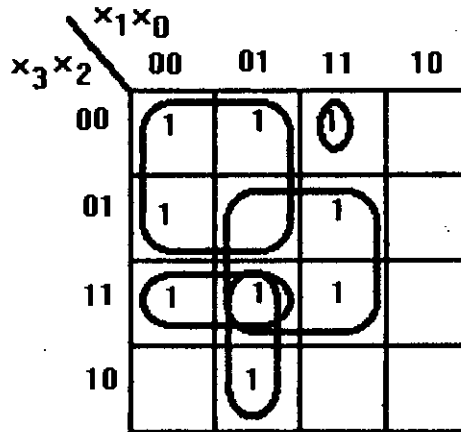
		$x_1 x_0$			
	$x_3 x_2$	00	01	11	10
00		b_0	b_1	b_3	b_2
01		b_4	b_5	b_7	b_6
11		b_{12}	b_{13}	b_{15}	b_{14}
10		b_8	b_9	b_{11}	b_{10}

which is similar to the K-Map except that entries are b-coefficient rather than truth table values. The use of map for the minimization is illustrated with the help of a four variable

function

$$f = 1 \oplus x_0 \oplus x_2 x_3 \oplus x_2 \oplus x_0 x_1 \oplus x_0 x_1 x_2 \oplus x_0 x_2 x_3 \oplus x_0 x_3 \oplus x_0 x_1 x_2 x_3$$

The map for the b coefficient would be



Rules for minimization :-

- 1) If the loop is within the true domain of any variable, that variable appears in the product in true form.
- 2) If the loop is within the false (complemented) domain of a variable, that variable does not appear in the product term.
- 3) If the loop spans both the true and false domains of a variable, that variable appears in the complemented form.

From the Map

$$f = \bar{x}_2 \bar{x}_0 \oplus x_2 x_0 \bar{x}_3 \bar{x}_1 \oplus x_1 x_0 \oplus x_3 x_2 \bar{x}_0 \oplus x_3 \bar{x}_2 x_0$$

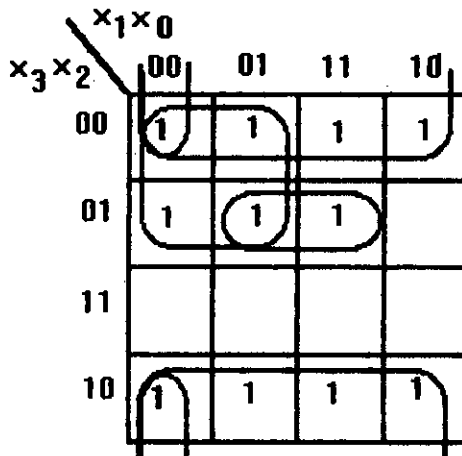
It is evident that for EXOR operation, the b=1 boxes on this map can only be included an odd number of times. On the other hand the b=0 boxes must not be included in loops, but may be included an even number of times.

When a positive polarity expression is entered on the map, a mixed/fixed polarity answer

is obtained. It is obvious that the map can be used to convert mixed polarity expansions into positive polarity. As an example for logic function

$$f = \bar{x}_2\bar{x}_0 \oplus x_2x_0\bar{x}_1 \oplus \bar{x}_3 \oplus \bar{x}_3\bar{x}_1\bar{x}_0$$

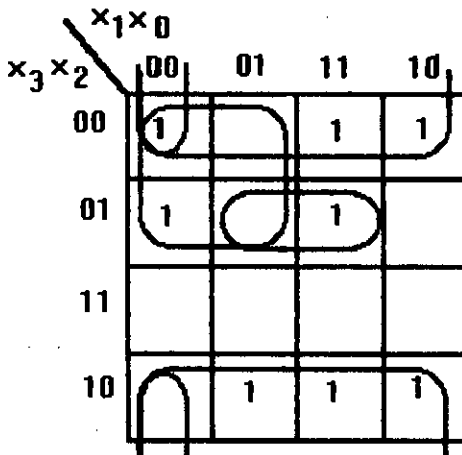
the RMC map would be



since b1,b5,b8 have even number of '1' so they are ignored and the resultant PPRME would be

$$f = 1 \oplus x_1 \oplus x_1x_0 \oplus x_2 \oplus x_2x_1x_0 \oplus x_3x_0 \oplus x_3x_1 \oplus x_3x_1x_0$$

Now for the reverse operation, i.e., PPRME to mixed polarity expression, we have RMC-MAP (i.e., b map)



From the above map we have

$$f = \bar{x}_2\bar{x}_0 \oplus x_2x_0\bar{x}_1 \oplus \bar{x}_3 \oplus \bar{x}_3\bar{x}_1\bar{x}_0$$

5.4.2 Tri-state Map Method:

Tri-state map is a tool developed to convert minterms into RM coefficient in fixed polarity and to minimize RM polynomials in mixed polarity [27]. Polarization is the process of converting a variable to a particular form, either true or complemented. Polarization of an n-variable function starts with its Karnaugh map. A folding technique is then used to remove the unwanted form of, or to polarize each folding. This process continues until all variables have been polarized. A transition or partially polarized map is obtained every time a variable is polarized. The transition map obtained from the polarization of the last variable is the RM-coefficient map. Every canonical product of an RM polynomial is represented by 1-entry in a cell in this map. The procedure of conversion from minterms to fixed polarity is summarized as follows:

- a) Draw the k-map for the Boolean expression.
- b) Decide on the required polarity of RM expansion.
- c) To polarize a variable x_i in positive (negative) polarity, fold the transition or karnaugh map along the boundary of x_i/\bar{x}_i so that the portion for $\bar{x}_i(x_i)$ is placed on top of the portion for $x_i(\bar{x}_i)$. Exclusive-OR the two portions and the resultant exclusive-OR values make up the $x_i(\bar{x}_i)$ portion of a transition map. The portion for $\bar{x}_i(x_i)$ remains unchanged.
- d) Replace the label of x_i which is 0(1) across the top or along the side of the map with '-'(' signifies the absence of x_i in the canonical products). The label of x_i which is

1(0) remains intact.

e) Repeat step c to step d for variables x_i , where $0 \leq i < n$

f) The 1s on the new map represent the product term of RM expression and the label of x_i signifies its complemented form or uncomplemented form.

In a transition map, a variable which has not yet been polarized is bipolar. It inherits from the karnaugh map the labels of 0 and 1. Its respective forms in a product are complemented or true. For a positively (negatively) polarized variable, it can occur only in true (complemented) form in a product or does not exist at all. Therefore there will be three different forms or states for the variables in a transition map: true, complemented and nonexistent. However, each variable can occur in only two of the three states.

Example:-

A four variable logic function is given in sum of products Boolean form as follows:

$$f(A,B,C,D) = \sum (0,2,3,5,6,9,10,12,15)$$

convert f into fixed polarity Reed-Muller expansion, in which A , C & D are in uncomplemented form and B is in complemented form.

Solution:

	CD			
AB	00	01	11	10
00	1		1	1
01		1		1
11	1		1	
10		1		1

(a)

	CD			
AB	00	01	11	10
-0	1		1	1
-1		1		1
11	1	1	1	1
10	1	1	1	

(b)

	CD			
AB	00	01	11	10
-0	1	1	1	
--		1		1
1-	1	1	1	1
10				1

(c)

	CD			
AB	-0	-1	11	10
-0	1	1		1
--		1	1	1
1-	1	1		
10				1

(d)

	CD			
AB	--	-1	11	1--
-0	1		1	1
--		1		1
1-	1			
10			1	1

(e)

Sequence of operation a → b → c → d → e

- Karnaugh map
- Transition map with A in +ve polarity
- Transition map with A in +ve and B in -ve polarity
- Transition map with A, C in +ve and B in -ve polarity
- RM-Coefficient map with A,C,D in +ve and B in -ve polarity.

Hence from the last map,

$$f = \bar{B} \oplus \bar{B} C D \oplus \bar{B} C \oplus D \oplus C \oplus A \oplus A \bar{B} C D \oplus A \bar{B} C$$

5.4.3 Map Folding Techniques:

This technique is similar to tristate map, the difference is the sequence of operation. The procedure is summarized as follows:

- Draw the k-map for the Boolean expression.
- Decide on the required polarity of RM expression.
- To eliminate the complemented form of any variable A, fold \bar{A} over A and EXOR the two portions to find the new content of the A section.
- Repeat for other variables starting from the last map generated in (c).

e) The 1s on the new map represent the product terms and if the labels of variables are 1s the corresponding variables are uncomplemented if true form is used. The variables would be complemented if complemented form is used.

The tristate-map example is illustrated below:

		CD			
AB		00	01	11	10
00	1			1	1
01			1		1
11	1			1	
10			1		1

(a)

		CD			
AB		00	01	11	10
00	1			1	1
01			1		1
11	1	1	1	1	1
10	1	1	1		

(b)

		CD			
AB		00	01	11	10
00	1			1	1
01	1	1	1		
11					1
10	1	1	1		

(c)

		CD			
AB		00	01	11	10
00	1			1	
01	1	1			1
11					1
10	1	1			1

(d)

		CD			
AB		00	01	11	10
00	1	1	1		
01	1		1	1	
11				1	1
10	1		1	1	

(e)

		CD			
AB		00	01	11	10
00		1			1
01	1			1	1
11				1	1
10	1				

(f)

- a. Karnaugh map
- b. K-map with eliminated \bar{A}
- c. " " " \bar{A}, \bar{B}
- d. " " " $\bar{A}, \bar{B}, \bar{C}$
- e. " for positive polarity RM expansion
- f. K-map with eliminated $\bar{A}, \bar{B}, \bar{C}, \bar{D}$

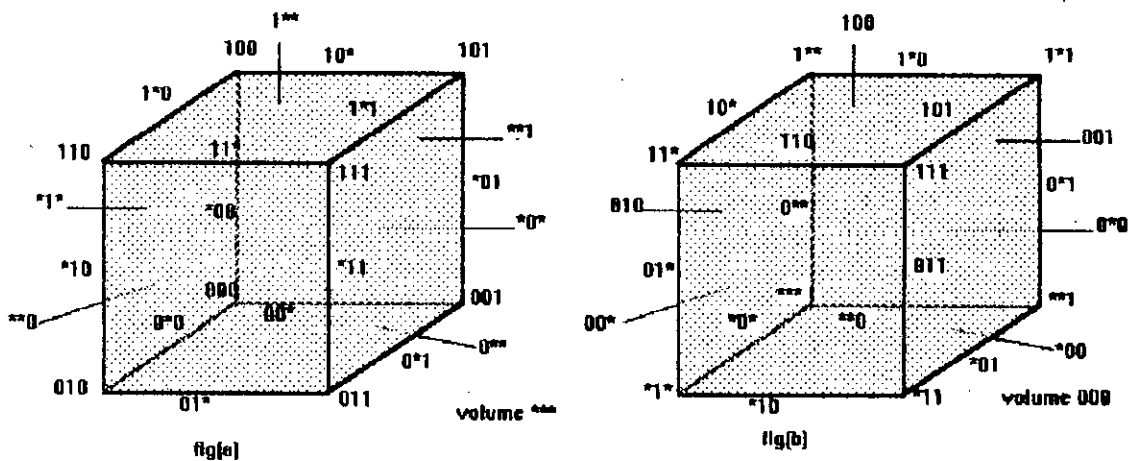
From the last map we can write

$$f = \bar{B} \oplus \bar{B} C D \oplus \bar{B} C \oplus D \oplus C \oplus A \oplus A \bar{B} C D \oplus A \bar{B} C$$

The criterion for minimization of any combinational logic circuit is the reduction of the number of products and also the number of literals. It is clear that the above mentioned minimization process for RM canonic expansion are impractical when functions of more than six-variables are dealt with. However, the concept of a tristate map can be extended to functions of any number of variables (see in [19]). Additionally we have to convert the function in the NOT/OR/NOR/NAND domain to AND-EXOR domain. It is preferable to devise an algorithm so that minimized AND-EXOR expression can be obtained directly from the truth table. In the next section such an algorithm is described step by step and its modification is implemented and studied.

5.4.4 Transformation from SOP Domain to ESOP Domain:

To describe this kind of Reed-Muller transform (RMT), consider the topological model of a 3-variable SOP as shown in (fig.a) and compare it with the corresponding representation of a 3-variable ESOP (fig.b) below



In the case of SOP representation, the 2^n vertices of n-dimensional hypercube corresponds to the 2^n minterms, whereas edges, surfaces and the volume represent product terms with n-1, n-2 and n-3 literals respectively. In contrast, in a positive polarity ESOP, vertices represent all the 2^n possible products of variables (including the constant 1). Combinations of edges, surfaces and volume are performed by mod-2 addition. Comparison of the 3^n combinations of the conventional symbols 1, 0 and * (standing for true, false and nonpresent literals of a product, respectively) of SOPs and ESOPs reveals a simple relationship;

i. Negated variables of SOPs are not present in the corresponding ESOP representation.

ii. Nonpresent variables in SOPs appear as negated variables in the associated ESOP.

iii. Uncomplemented variables remain same in both domains.

In other words the exchange of zeros and asterisks of the true cubes (i.e., minterms or products) transforms them into the RM domain.

Example: Transform the following function to the PPRME

x_2	x_1	x_0	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Solution:

Replacing zeros by asterisks we have the ON terms

w_2	w_1	w_0	f
*	*	1	1
*	1	*	1
1	1	*	1
1	1	1	1

w_2, w_1, w_0 denote the variables in the RM domain.

* * 1 this cube cover 001, 011, 101, 111

* 1 * this cube cover 010, 011, 110, 111

1 1 * this cube cover 110, 111

1 1 1 this cube cover 111

So in the RM domain output would be zero if the above cubes do not cover the minterm or cover the minterm in even number of times. Output would be one if the above cubes cover a minterm in odd number of times.

So the function in the RM domain would be

w_2	w_1	w_0	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Hence, $f = x_0 + x_1 + x_2x_0$

Method of simplification:

1. To simplify the following function in ESOP domain we make cover(s) disjoint.

x_2	x_1	x_0	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Accordingly the ON terms and the corresponding K-Map would be

x_2 x_1 x_0
 0 1 0
 1 0 0
 1 * 1

		x_1x_0			
		00	01	11	10
x_2	0				1
	1	1	1	1	

2. To obtain a criterion for the choice of a near optimum RM polarity for each variable x_i , a value k_i is generated in the following way:

Let d_i denote the number of asterisks in the i th cube. Further, let for the j th cube and the i th variable

$$a_{ij} = \begin{cases} 1 & \text{if } x_i = 1 \\ 0 & \text{if } x_i = * \\ -1 & \text{if } x_i = 0 \end{cases}$$

Then we define

$$k_i = \sum_{j=0}^r a_{ij} 2^{d_j}$$

Depending on the sign of the k_i , the fixed polarity RMT of a disjoint cover is performed by the assignments

$$\begin{aligned} x_i &\rightarrow w_i \\ 1 &\rightarrow 1 \\ * &\rightarrow 0 \\ 0 &\rightarrow * \quad \text{for } k_i \geq 0 \end{aligned}$$

and

$$\begin{aligned} x_i &\rightarrow w_i && \text{SOP Polarized ESOP} \\ 1 &\rightarrow * && (\text{i.e. } 1 \rightarrow 0 \rightarrow *) \\ * &\rightarrow 0 && (\text{i.e. } * \rightarrow * \rightarrow 0) \\ 0 &\rightarrow 1 && (\text{i.e. } 0 \rightarrow 1 \rightarrow 1) \end{aligned}$$

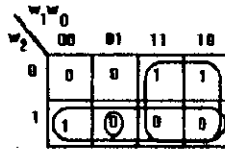
for $k_i < 0$

Application of the above rules leads to the following values of k_i : $k_0=0$, $k_1=0$, $k_2=2$. So the variables would be in uncomplemented form and the polarized ON terms would be

x_2	x_1	x_0
0	1	0
1	0	0
1	*	1

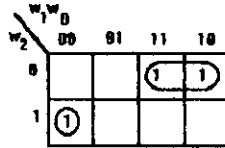
3. The transformation into the RM domain is carried out by exchanging '*' and '0'. We have the ON terms and the corresponding K-Map

w_2	w_1	w_0
*	1	*
1	*	*
1	0	1



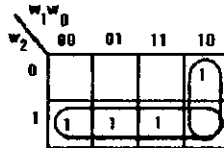
4. Forming the disjoint RM domain cubes we have minimum product RMC

w_2	w_1	w_0
0	1	*
1	0	0



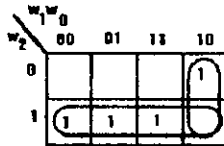
5. A polarized minimum product representation is obtained by inverse RMT, i.e., by exchanging '*' and '0' again. The resultant ON terms and K-Map would be

x_2	x_1	x_0
*	1	0
1	*	*



6. Converting to positive polarity results in a (near) minimum EXOR cover as illustrated below

x_2	x_1	x_0
*	1	0
1	*	*



So $f = x_2 \oplus x_1 \bar{x}_0$

5.5 Mlynarovic's Method for Minimization:

It is a technique for minimization of many-variable problems. Algorithmic complexity of a minimizing method increases proportionally with the number of input combinations, i.e., 2^n , where n is the number of variables. With the increase number of variables, the memory requirement also increases and varies with different techniques. Mlynarovic method is a matrix formulated algorithm for finding minimal RMC forms. But such an algorithm is not efficient since the growth of n results in rapid growth of the size of matrix.

Reed-Muller Canonic (RMC) expansion of n variables function f is given by

$$f(x_n, x_{n-1}, \dots, x_2, x_1) = b_0 \oplus b_1 x_1^* \oplus b_2 x_2^* \oplus b_3 x_1^* x_2^* \oplus \dots \oplus b_{2^n-1} x_1^* x_2^* \dots x_n^*$$

where x^* represents x or \bar{x}

Definition-1:

If $X = (x_n^*, x_{n-1}^*, \dots, x_1^*)$ is a vector polarity and $P = (p_n, p_{n-1}, \dots, p_2, p_1)$

is a coefficient polarity, then vector polarity X can be defined as follows:

$$X = (x_n \oplus p_n, x_{n-1} \oplus p_{n-1}, \dots, x_1 \oplus p_1)$$

where p_i is 0 or 1.

If we have 2^n different types of vector polarity, we will get 2^n different types of Reed-

Muller Canonic expansions. The RMC forms with different values of p are described as follows:

when $p=0$:

$$f(x_n, x_{n-1}, \dots, x_1) = a_{0\dots 00} \oplus a_{0\dots 01}x_1 \oplus a_{0\dots 10}x_2 \oplus a_{0\dots 11}x_1x_2 \oplus \dots \oplus a_{1\dots 11}x_1x_2\dots x_n$$

when $p=1$:

$$f(x_n, x_{n-1}, \dots, \bar{x}_1) = a_{0\dots 00} \oplus a_{0\dots 01}\bar{x}_1 \oplus a_{0\dots 10}x_2 \oplus a_{0\dots 11}\bar{x}_1x_2 \oplus \dots \oplus a_{1\dots 11}\bar{x}_1x_2\dots x_n$$

when $p=2$:

$$f(x_n, x_{n-1}, \dots, \bar{x}_2, x_1) = a_{0\dots 00} \oplus a_{0\dots 01}x_1 \oplus a_{0\dots 10}\bar{x}_2 \oplus a_{0\dots 11}x_1\bar{x}_2 \oplus \dots \oplus a_{1\dots 11}x_1x_2\dots x_n$$

when $p=2^n-1$:

$$f(\bar{x}_n, \bar{x}_{n-1}, \dots, \bar{x}_1) = a_{0\dots 00} \oplus a_{0\dots 01}\bar{x}_1 \oplus a_{0\dots 10}\bar{x}_2 \oplus a_{0\dots 11}\bar{x}_1\bar{x}_2 \oplus \dots \oplus a_{1\dots 11}\bar{x}_1\bar{x}_2\dots \bar{x}_n$$

a_i is coefficient with binary expansion

$$i = i_n i_{n-1} \dots i_1$$

which indicates which product terms are presented in the expansion.

Definition-2:

If A_p is a vector of coefficients of Reed-Muller Canonic expansion and p is a coefficient polarity of Boolean function $f(x_n, x_{n-1}, x_{n-2}, \dots, x_1)$ then

$$A_p = (a_0, a_1, a_2, \dots, a_{2^n-1})$$

Definition-3:

Reed-Muller Canonic expansion of Boolean function $f(x_n, x_{n-1}, \dots, x_1)$ is minimal if its A_p (vector of coefficients) is obtained by least number of 1s.

So firstly, all A_p are calculated for polarity coefficient $p(0 \leq p < 2^n)$ and then A_p with least number of 1s is selected.

Lemma-1:

If A_p is a vector of coefficients of Boolean function $f(x_n, x_{n-1}, \dots, x_1)$ and vector of polarity $p=0$, then

$$A_0 = S^n V_0$$

where V_0 is a canonical vector of Boolean function $f(x_n, x_{n-1}, \dots, x_1)$

$$V_0 = (f(0), f(1), \dots, f(2^n-1)).$$

S^n is a Boolean matrix of size $2^n \times 2^n$ which is recursively defined as:

$$S^n = \begin{bmatrix} S^{n-1} & 0 \\ S^{n-1} & S^{n-1} \end{bmatrix} \quad S^0 = 1$$

where $n =$ total number of input variables

Lemma-2:

If A_p is a vector of coefficients of Boolean function $f(x_n, x_{n-1}, \dots, x_1)$ and p is a coefficient polarity, then

$$A_p = S^n \cdot V_p$$

where p is polarity coefficient ($0 \leq p \leq 2^n - 1$)

n = Total number of input-variables

V_p = Permutation of Canonical Vector V with respect to coefficient p .

$$V_p = (f(0 \oplus p), f(1 \oplus p), \dots, f((2^n - 1) \oplus p))$$

where $f(i \oplus p) = f(i_n \oplus p_n, i_{n-1} \oplus p_{n-1}, \dots, i_1 \oplus p_1)$ and $0 \leq p \leq 2^n - 1$

Symbol "." represents Boolean multiplication, i.e., multiplication which is combined by logic operations AND and EXOR.

The following example explains the Mlynarovic's method for minimization.

x_2	x_1	f
0	0	1
0	1	0
1	0	1
1	1	1

Total number of inputs $n=2$ and $0 \leq p \leq 2^2 - 1$, i.e., $0 \leq p \leq 3$. According to Lemma-2, we can calculate

$$\begin{aligned} \text{For } p=0, \quad V_0 &= (f(0 \oplus 0), f(1 \oplus 0), f(2 \oplus 0), f(3 \oplus 0)) \\ &= (f(0), f(1), f(2), f(3)) \\ &= (1, 0, 1, 1) \end{aligned}$$

$$\begin{aligned} \text{For } p=1 \quad V_1 &= (f(0 \oplus 1), f(1 \oplus 1), f(2 \oplus 1), f(3 \oplus 1)) \\ &= (f(1), f(0), f(3), f(2)) \\ &= (0, 1, 1, 1) \end{aligned}$$

$$\begin{aligned} \text{For } p=2 \quad V_2 &= (f(0 \oplus 2), f(1 \oplus 2), f(2 \oplus 2), f(3 \oplus 2)) \\ &= (f(2), f(3), f(0), f(1)) \\ &= (1, 1, 1, 0) \end{aligned}$$

$$\begin{aligned} \text{For } p=3 \quad V_3 &= (f(0 \oplus 3), f(1 \oplus 3), f(2 \oplus 3), f(3 \oplus 3)) \\ &= (f(3), f(2), f(1), f(0)) \\ &= (1, 1, 0, 1) \end{aligned}$$

By using Lemma-1

$$S^1 = \begin{bmatrix} S^0 & 0 \\ S^0 & S^0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

$$S^2 = \begin{bmatrix} S^1 & 0 \\ S^1 & S^1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Now,

$$\begin{aligned} A_0 &= S^2 V_0 \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} \end{aligned}$$

$$\begin{aligned} A_1 &= S^2 V_1 \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} \end{aligned}$$

$$\begin{aligned} A_2 &= S^2 V_2 \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \end{aligned}$$

$$\begin{aligned} A_3 &= S^2 V_3 \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} \end{aligned}$$

According to definition-3, our RMC form must be formed from A_2 because A_2 obtains least number of 1s. To calculate the polarity of input variables of the minimal canonical forms, we need to convert the value of p , i.e., 2 into a binary number which also represents the values of the input variables, i.e.,

$$p = 2 = \begin{matrix} x_2 & x_1 \\ 1 & 0 \end{matrix}$$

where '1' indicates that x_2 must be in complemented form and '0' indicates that x_1 must be in true form in the minimal RMC canonical expression. So the minimal canonical form of the given function is

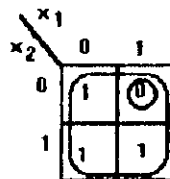
$$f = \begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_2 & x_1 \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$$

$$= 1 \oplus 0 \cdot x_1 \oplus 0 \cdot x_2 \oplus 1 \cdot \bar{x}_2 x_1$$

$$= 1 \oplus \bar{x}_2 x_1$$

To verify this expression, we can put the above function in the Karnaugh Map and looping the 1s and 0s according to the above expression we get

K-Map:



1s in the K-Map is covered by odd number of loops and 0s are covered by even number

of loops. So the above expression is incorrect, i.e., $f(x_2, x_1) = 1 \oplus \bar{x}_2 x_1$ for the given function.

5.6 Modified Algorithm:

Polarity Function:

$X=(x_n, x_{n-1}, \dots, x_1)$ polarity vector and $P=(p_n, p_{n-1}, \dots, p_1)$ polarity coefficients

Definition-1:

If A_p is a vector of coefficients of RMC for definite Boolean function with polarity p , then

$$A_p = (a_0, a_1, \dots, a_{2^n-1})$$

Definition-2:

An RMC form of a Boolean function is minimal if its A_p (vector of coefficients) obtains least number of "1".

So firstly, we must calculate all A_p for polarity coefficient p ($0 \leq p \leq 2^n-1$) and then we can select A_p with least number of "1".

Definition-3:

M^k is a vector of components for all $1 \leq k \leq n$ which obtains $2^n/2^k$ segments. Each of this segment starts with zero(s) and then is followed by one(s). The total number of zero(s) or one(s) in each segment is 2^{k-1} .

For example: For $n=3$

$$M^1=(0,1,0,1,0,1,0,1)$$

$$M^2=(0,0,1,1,0,0,1,1)$$

$$M^3=(0,0,0,0,1,1,1,1)$$

Lemma-1:

If $f(x_n, x_{n-1}, \dots, x_1)$ is an n-variable Boolean function and p ($0 \leq p \leq 2^n-1$) is polarity coefficient, then vector of coefficients is defined as $A_p = B_p^n$

where B_p^n is a vector which obtains i components ($0 \leq i \leq 2^n-1$)

The three main mathematical notations for this algorithm are as follows:

$$B_p^n(i) = B_p^{n-1}(i) \oplus (M^n(i) \cdot B_p^{n-1}(i \oplus 2^{n-1})) \dots \dots \dots (5.1)$$

$$B_p^0(i) = V_p(i) = f(i \oplus p) \dots \dots \dots (5.2)$$

$$B_p^0(i \oplus 2^{k-1}) = V_p(i \oplus 2^{k-1} \oplus p) \dots \dots \dots (5.3)$$

for $1 \leq k \leq n$

As for example: If $n=3$ and $p=0$, then

$$B_0^1(i) = B_0^0(i) \oplus (M^1(i) \cdot B_0^0(i \oplus 2^0))$$

$$B_0^2(i) = B_0^1(i) \oplus (M^2(i) \cdot B_0^1(i \oplus 2^1))$$

$$B_0^3(i) = B_0^2(i) \oplus (M^3(i) \cdot B_0^2(i \oplus 2^2))$$

Example-1: Let us consider a function $f(x_3, x_2, x_1)$, the truth table of this function is given below:

x_3	x_2	x_1	f
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

M^k = vector of components.

where $1 \leq k \leq n$ and $n=3$ (n =total number of input-variables)

If $k=1$, then total number of segments $=2^n/2^k=2^3/2^1=4$

$$M^1=(0,1,0,1,0,1,0,1)$$

If $k=2$, then total number of segments $=2^n/2^k=2^3/2^2=2$

$$M^2=(0,0,1,1,0,0,1,1)$$

If $k=3$, then total number of segments $=2^n/2^k=2^3/2^3=1$

$$M^3=(0,0,0,0,1,1,1,1)$$

Now, If $p=0$, then

$$B^0(i)=V_0(i)=f(i \oplus 0), 0 \leq i \leq 2^n-1$$

$$\begin{aligned} V_0(i) &= (f(0), f(1), f(2), f(3), f(4), f(5), f(6), f(7)) \\ &= (1, 1, 0, 0, 1, 0, 1, 1) \end{aligned}$$

If $p=0$ and $n=1$, then

$$B^1_0(i)=B^0_0(i) \oplus (M^1(i).B^0_0(i \oplus 1))$$

$$B^1_0(0)=1 \oplus (0.1) = 1$$

$$B^1_0(1)=1 \oplus (1.1) = 0$$

$$B^1_0(2)=0 \oplus (0.0) = 0$$

$$B^1_0(3)=0 \oplus (1.0) = 0$$

$$B^1_0(4)=1 \oplus (0.0) = 1$$

$$B^1_0(5)=0 \oplus (1.1) = 1$$

$$B^1_0(6)=1 \oplus (0.1) = 1$$

$$B^1_0(7)=1 \oplus (1.1) = 0$$

So, $B^1_0 = (1, 0, 0, 0, 1, 1, 1, 0)$

If $p=0$ and $n=2$, then

$$B^2_0(i) = B^1_0(i) \oplus (M^2(i) \cdot B^1_0(i \oplus 2))$$

$$B^2_0(0) = 1 \oplus (0.0) = 1$$

$$B^2_0(1) = 0 \oplus (0.0) = 0$$

$$B^2_0(2) = 0 \oplus (1.1) = 1$$

$$B^2_0(3) = 0 \oplus (1.0) = 0$$

$$B^2_0(4) = 1 \oplus (0.1) = 1$$

$$B^2_0(5) = 1 \oplus (0.0) = 1$$

$$B^2_0(6) = 1 \oplus (1.1) = 0$$

$$B^2_0(7) = 0 \oplus (1.1) = 1$$

So, $B^2_0 = (1,0,1,0,1,1,0,1)$

If $p=0$ and $n=3$, then

$$B^3_0(i) = B^2_0(i) \oplus (M^3(i) \cdot B^2_0(i \oplus 4))$$

$$B^3_0(0) = 1 \oplus (0.1) = 1$$

$$B^3_0(1) = 0 \oplus (0.1) = 0$$

$$B^3_0(2) = 1 \oplus (0.0) = 1$$

$$B^3_0(3) = 0 \oplus (0.1) = 0$$

$$B^3_0(4) = 1 \oplus (1.1) = 0$$

$$B^3_0(5) = 1 \oplus (1.0) = 1$$

$$B^3_0(6) = 0 \oplus (1.1) = 1$$

$$B^3_0(7) = 1 \oplus (1.0) = 1$$

So, $B^3_0 = (1,0,1,0,1,1,0,1) = A_0$

Now, If $p=1$, then

$$B^0_1(i) = V_1(i) = f(i \oplus 1), 0 \leq i \leq 2^n - 1$$

$$\begin{aligned} V_1(i) &= (f(1), f(0), f(3), f(2), f(5), f(4), f(7), f(6)) \\ &= (1, 1, 0, 0, 0, 1, 1, 1) \end{aligned}$$

If $p=1$ and $n=1$, then

$$B^1_1(i) = B^0_1(i) \oplus (M^1(i) \cdot B^0_1(i \oplus 1))$$

$$B^1_1(0) = 1 \oplus (0.1) = 1$$

$$B^1_1(1) = 1 \oplus (1.1) = 0$$

$$B^1_1(2) = 0 \oplus (0.0) = 0$$

$$B^1_1(3) = 0 \oplus (1.0) = 0$$

$$B^1_1(4) = 0 \oplus (0.1) = 0$$

$$B^1_1(5) = 1 \oplus (1.0) = 1$$

$$B^1_1(6) = 1 \oplus (0.1) = 1$$

$$B^1_1(7) = 1 \oplus (1.1) = 0$$

So, $B^1_1 = (1,0,0,0,0,1,1,0)$

If $p=1$ and $n=2$, then

$$B^2_1(i) = B^1_1(i) \oplus (M^2(i).B^1_1(i \oplus 2))$$

$$B^2_1(0) = 1 \oplus (0.0) = 1$$

$$B^2_1(1) = 0 \oplus (0.0) = 0$$

$$B^2_1(2) = 0 \oplus (1.1) = 1$$

$$B^2_1(3) = 0 \oplus (1.0) = 0$$

$$B^2_1(4) = 0 \oplus (0.1) = 0$$

$$B^2_1(5) = 1 \oplus (0.0) = 1$$

$$B^2_1(6) = 1 \oplus (1.0) = 1$$

$$B^2_1(7) = 0 \oplus (1.1) = 1$$

So, $B^2_1 = (1,0,1,0,0,1,1,1)$

If $p=1$ and $n=3$, then

$$B^3_1(i) = B^2_1(i) \oplus (M^3(i).B^2_1(i \oplus 4))$$

$$B^3_1(0) = 1 \oplus (0.0) = 1$$

$$B^3_1(1) = 0 \oplus (0.1) = 0$$

$$B^3_1(2) = 1 \oplus (0.1) = 1$$

$$B^3_1(3) = 0 \oplus (0.1) = 0$$

$$B_1^3(4) = 0 \oplus (1.1) = 1$$

$$B_1^3(5) = 1 \oplus (1.0) = 1$$

$$B_1^3(6) = 1 \oplus (1.1) = 0$$

$$B_1^3(7) = 1 \oplus (1.0) = 1$$

$$\text{So, } B_1^3 = (1,0,1,0,1,1,0,1) = A_1$$

Now, If $p=2$, then

$$B_2^0(i) = V_2(i) = f(i \oplus 2), 0 \leq i \leq 2^n - 1$$

$$\begin{aligned} V_1(i) &= (f(2), f(3), f(0), f(1), f(6), f(7), f(4), f(5)) \\ &= (0, 0, 1, 1, 1, 1, 1, 0) \end{aligned}$$

If $p=2$ and $n=1$, then

$$B_2^1(i) = B_2^0(i) \oplus (M^1(i) \cdot B_2^0(i \oplus 1))$$

$$B_2^1(0) = 0 \oplus (0.0) = 0$$

$$B_2^1(1) = 0 \oplus (1.0) = 0$$

$$B_2^1(2) = 1 \oplus (0.1) = 1$$

$$B_2^1(3) = 1 \oplus (1.1) = 0$$

$$B_2^1(4) = 1 \oplus (0.1) = 1$$

$$B_2^1(5) = 1 \oplus (1.1) = 0$$

$$B_2^1(6) = 1 \oplus (0.0) = 1$$

$$B_2^1(7) = 0 \oplus (1.1) = 1$$

So, $B_2^1 = (0,0,1,0,1,0,1,1)$

If $p=2$ and $n=2$, then

$$B_2^2(i) = B_2^1(i) \oplus (M^2(i).B_2^1(i \oplus 2))$$

$$B_2^2(0) = 0 \oplus (0.1) = 0$$

$$B_2^2(1) = 0 \oplus (0.0) = 0$$

$$B_2^2(2) = 1 \oplus (1.0) = 1$$

$$B_2^2(3) = 0 \oplus (1.0) = 0$$

$$B_2^2(4) = 1 \oplus (0.1) = 1$$

$$B_2^2(5) = 0 \oplus (0.1) = 0$$

$$B_2^2(6) = 1 \oplus (1.1) = 0$$

$$B_2^2(7) = 1 \oplus (1.0) = 1$$

So, $B_2^2 = (0,0,1,0,1,0,0,1)$

If $p=2$ and $n=3$, then

$$B_2^3(i) = B_2^2(i) \oplus (M^3(i).B_2^2(i \oplus 4))$$

$$B_2^3(0) = 0 \oplus (0.1) = 0$$

$$B_2^3(1) = 0 \oplus (0.0) = 0$$

$$B_2^3(2) = 1 \oplus (0.0) = 1$$

$$B_2^3(3) = 0 \oplus (0.1) = 0$$

$$B_2^3(4) = 1 \oplus (1.0) = 1$$

$$B_2^3(5) = 0 \oplus (1.0) = 0$$

$$B_2^3(6) = 0 \oplus (1.1) = 1$$

$$B_2^3(7) = 1 \oplus (1.0) = 1$$

$$\text{So, } B_2^3 = (0,0,1,0,1,0,1,1) = A_2$$

Now, If $p=3$, then

$$B_3^0(i) = V_3(i) = f(i \oplus 3), \quad 0 \leq i \leq 2^n - 1$$

$$\begin{aligned} V_3(i) &= (f(3), f(2), f(1), f(0), f(7), f(6), f(5), f(4)) \\ &= (0,0,1,1,1,1,0,1) \end{aligned}$$

If $p=3$ and $n=1$, then

$$B_3^1(i) = B_3^0(i) \oplus (M^1(i) \cdot B_3^0(i \oplus 1))$$

$$B_3^1(0) = 0 \oplus (0.0) = 0$$

$$B_3^1(1) = 0 \oplus (1.0) = 0$$

$$B_3^1(2) = 1 \oplus (0.1) = 1$$

$$B_3^1(3) = 1 \oplus (1.1) = 0$$

$$B_3^1(4) = 1 \oplus (0.1) = 1$$

$$B_3^1(5) = 1 \oplus (1.1) = 0$$

$$B_3^1(6) = 0 \oplus (0.1) = 0$$

$$B_3^1(7) = 1 \oplus (1.0) = 1$$

So, $B^1_3 = (0,0,1,0,1,0,0,1)$

If $p=3$ and $n=2$, then

$$B^2_3(i) = B^1_3(i) \oplus (M^2(i).B^1_3(i \oplus 2))$$

$$B^2_3(0) = 0 \oplus (0.1) = 0$$

$$B^2_3(1) = 0 \oplus (0.0) = 0$$

$$B^2_3(2) = 1 \oplus (1.0) = 1$$

$$B^2_3(3) = 0 \oplus (1.0) = 0$$

$$B^2_3(4) = 1 \oplus (0.0) = 1$$

$$B^2_3(5) = 0 \oplus (0.1) = 0$$

$$B^2_3(6) = 0 \oplus (1.1) = 1$$

$$B^2_3(7) = 1 \oplus (1.0) = 1$$

So, $B^2_3 = (0,0,1,0,1,0,1,1)$

If $p=3$ and $n=3$, then

$$B^3_3(i) = B^2_3(i) \oplus (M^3(i).B^2_3(i \oplus 4))$$

$$B^3_3(0) = 0 \oplus (0.1) = 0$$

$$B^3_3(1) = 0 \oplus (0.0) = 0$$

$$B^3_3(2) = 1 \oplus (0.1) = 1$$

$$B^3_3(3) = 0 \oplus (0.1) = 0$$

$$B^3_3(4) = 1 \oplus (1.0) = 1$$

$$B^3_3(5) = 0 \oplus (1.0) = 0$$

$$B^3_3(6) = 1 \oplus (1.1) = 0$$

$$B^3_3(7) = 1 \oplus (1.0) = 1$$

$$\text{So, } B^3_3 = (0,0,1,0,1,0,0,1) = A_3$$

Now, If $p=4$, then

$$B^0_4(i) = V_4(i) = f(i \oplus 4), \quad 0 \leq i \leq 2^4 - 1$$

$$\begin{aligned} V_4(i) &= (f(4), f(5), f(6), f(7), f(0), f(1), f(2), f(3)) \\ &= (1, 0, 1, 1, 1, 1, 0, 0) \end{aligned}$$

If $p=4$ and $n=1$, then

$$B^1_4(i) = B^0_4(i) \oplus (M^1(i) \cdot B^0_4(i \oplus 1))$$

$$B^1_4(0) = 1 \oplus (0.0) = 1$$

$$B^1_4(1) = 0 \oplus (1.1) = 1$$

$$B^1_4(2) = 1 \oplus (0.1) = 1$$

$$B^1_4(3) = 1 \oplus (1.1) = 0$$

$$B^1_4(4) = 1 \oplus (0.1) = 1$$

$$B^1_4(5) = 1 \oplus (1.1) = 0$$

$$B^1_4(6) = 0 \oplus (0.0) = 0$$

$$B^1_4(7) = 0 \oplus (1.0) = 0$$

So, $B^1_4 = (1,1,1,0,1,0,0,0)$

If $p=4$ and $n=2$, then

$$B^2_4(i) = B^1_4(i) \oplus (M^2(i).B^1_4(i \oplus 2))$$

$$B^2_4(0) = 1 \oplus (0.1) = 1$$

$$B^2_4(1) = 1 \oplus (0.0) = 1$$

$$B^2_4(2) = 1 \oplus (1.1) = 0$$

$$B^2_4(3) = 0 \oplus (1.1) = 1$$

$$B^2_4(4) = 1 \oplus (0.0) = 1$$

$$B^2_4(5) = 0 \oplus (0.0) = 0$$

$$B^2_4(6) = 0 \oplus (1.1) = 1$$

$$B^2_4(7) = 0 \oplus (1.0) = 0$$

So, $B^2_4 = (1,1,0,1,1,0,1,0)$

If $p=4$ and $n=3$, then

$$B^3_4(i) = B^2_4(i) \oplus (M^3(i).B^2_4(i \oplus 4))$$

$$B^3_4(0) = 1 \oplus (0.1) = 1$$

$$B^3_4(1) = 1 \oplus (0.0) = 1$$

$$B^3_4(2) = 0 \oplus (0.1) = 0$$

$$B^3_4(3) = 1 \oplus (0.0) = 1$$

$$B^3_4(4) = 1 \oplus (1.1) = 0$$

$$B^3_4(5) = 0 \oplus (1.1) = 1$$

$$B^3_4(6) = 1 \oplus (1.0) = 1$$

$$B^3_4(7) = 0 \oplus (1.1) = 1$$

$$\text{So, } B^3_4 = (1,1,0,1,0,1,1,1) = A_4$$

Now, If $p=5$, then

$$B^0_5(i) = V_5(i) = f(i \oplus 5), 0 \leq i \leq 2^5 - 1$$

$$\begin{aligned} V_5(i) &= (f(5), f(4), f(7), f(6), f(1), f(0), f(3), f(2)) \\ &= (0, 1, 1, 1, 1, 1, 0, 0) \end{aligned}$$

If $p=5$ and $n=1$, then

$$B^1_5(i) = B^0_5(i) \oplus (M^1(i).B^0_5(i \oplus 1))$$

$$B^1_5(0) = 0 \oplus (0.1) = 0$$

$$B^1_5(1) = 1 \oplus (1.0) = 1$$

$$B^1_5(2) = 1 \oplus (0.1) = 1$$

$$B^1_5(3) = 1 \oplus (1.1) = 0$$

$$B^1_5(4) = 1 \oplus (0.1) = 1$$

$$B^1_5(5) = 1 \oplus (1.1) = 0$$

$$B^1_5(6) = 0 \oplus (0.0) = 0$$

$$B^1_5(7) = 0 \oplus (1.0) = 0$$

So, $B^1_5 = (0,1,1,0,1,0,0,0)$

If $p=5$ and $n=2$, then

$$B^2_5(i) = B^1_5(i) \oplus (M^2(i) \cdot B^1_5(i \oplus 2))$$

$$B^2_5(0) = 0 \oplus (0.1) = 0$$

$$B^2_5(1) = 1 \oplus (0.0) = 1$$

$$B^2_5(2) = 1 \oplus (1.0) = 1$$

$$B^2_5(3) = 0 \oplus (1.1) = 1$$

$$B^2_5(4) = 1 \oplus (0.0) = 1$$

$$B^2_5(5) = 0 \oplus (0.0) = 0$$

$$B^2_5(6) = 0 \oplus (1.1) = 1$$

$$B^2_5(7) = 0 \oplus (1.0) = 0$$

So, $B^2_5 = (0,1,1,1,1,0,1,0)$

If $p=5$ and $n=3$, then

$$B^3_5(i) = B^2_5(i) \oplus (M^3(i) \cdot B^2_5(i \oplus 4))$$

$$B^3_5(0) = 0 \oplus (0.1) = 0$$

$$B^3_5(1) = 1 \oplus (0.0) = 1$$

$$B^3_5(2) = 1 \oplus (0.1) = 1$$

$$B^3_5(3) = 1 \oplus (0.0) = 1$$

$$B^3_3(4) = 1 \oplus (1.0) = 1$$

$$B^3_3(5) = 0 \oplus (1.1) = 1$$

$$B^3_3(6) = 1 \oplus (1.1) = 0$$

$$B^3_3(7) = 0 \oplus (1.1) = 1$$

So, $B^3_3 = (0,1,1,1,1,1,0,1) = A_3$

Now, If $p=6$, then

$$B^0_6(i) = V_6(i) = f(i \oplus 6), 0 \leq i \leq 2^n - 1$$

$$\begin{aligned} V_6(i) &= (f(6), f(7), f(4), f(5), f(2), f(3), f(0), f(1)) \\ &= (1, 1, 1, 0, 0, 0, 1, 1) \end{aligned}$$

If $p=6$ and $n=1$, then

$$B^1_6(i) = B^0_6(i) \oplus (M^1(i), B^0_6(i \oplus 1))$$

$$B^1_6(0) = 1 \oplus (0.1) = 1$$

$$B^1_6(1) = 1 \oplus (1.1) = 0$$

$$B^1_6(2) = 1 \oplus (0.0) = 1$$

$$B^1_6(3) = 0 \oplus (1.1) = 1$$

$$B^1_6(4) = 0 \oplus (0.0) = 0$$

$$B^1_6(5) = 0 \oplus (1.0) = 0$$

$$B^1_6(6) = 1 \oplus (0.1) = 1$$

$$B^1_6(7) = 1 \oplus (1.1) = 0$$

So, $B^1_6 = (1,0,1,1,0,0,1,0)$

If $p=6$ and $n=2$, then

$$B^2_6(i) = B^1_6(i) \oplus (M^2(i).B^1_6(i \oplus 2))$$

$$B^2_6(0) = 1 \oplus (0.1) = 1$$

$$B^2_6(1) = 0 \oplus (0.1) = 0$$

$$B^2_6(2) = 1 \oplus (1.1) = 0$$

$$B^2_6(3) = 1 \oplus (1.0) = 1$$

$$B^2_6(4) = 0 \oplus (0.1) = 0$$

$$B^2_6(5) = 0 \oplus (0.0) = 0$$

$$B^2_6(6) = 1 \oplus (1.0) = 1$$

$$B^2_6(7) = 0 \oplus (1.0) = 0$$

So, $B^2_6 = (1,0,0,1,0,0,1,0)$

If $p=6$ and $n=3$, then

$$B^3_6(i) = B^2_6(i) \oplus (M^3(i).B^2_6(i \oplus 4))$$

$$B^3_6(0) = 1 \oplus (0.0) = 1$$

$$B^3_6(1) = 0 \oplus (0.0) = 0$$

$$B^3_6(2) = 0 \oplus (0.1) = 0$$

$$B^3_6(3) = 1 \oplus (0.0) = 1$$

$$B_6^3(4) = 0 \oplus (1.1) = 1$$

$$B_6^3(5) = 0 \oplus (1.0) = 0$$

$$B_6^3(6) = 1 \oplus (1.0) = 1$$

$$B_6^3(7) = 0 \oplus (1.1) = 1$$

So, $B_6^3 = (1,0,0,1,1,0,1,1) = A_6$

Now, If $p=7$, then

$$B_7^0(i) = V_7(i) = f(i \oplus 7), \quad 0 \leq i \leq 2^7-1$$

$$\begin{aligned} V_7(i) &= (f(7), f(6), f(5), f(4), f(3), f(2), f(1), f(0)) \\ &= (1, 1, 0, 1, 0, 0, 1, 1) \end{aligned}$$

If $p=7$ and $n=1$, then

$$B_7^1(i) = B_7^0(i) \oplus (M^1(i).B_7^0(i \oplus 1))$$

$$B_7^1(0) = 1 \oplus (0.1) = 1$$

$$B_7^1(1) = 1 \oplus (1.1) = 0$$

$$B_7^1(2) = 0 \oplus (0.1) = 0$$

$$B_7^1(3) = 1 \oplus (1.0) = 1$$

$$B_7^1(4) = 0 \oplus (0.0) = 0$$

$$B_7^1(5) = 0 \oplus (1.0) = 0$$

$$B_7^1(6) = 1 \oplus (0.1) = 1$$

$$B_7^1(7) = 1 \oplus (1.1) = 0$$

So, $B^1_7 = (1,0,0,1,0,0,1,0)$

If $p=7$ and $n=2$, then

$$B^2_7(i) = B^1_7(i) \oplus (M^2(i).B^1_7(i \oplus 2))$$

$$B^2_7(0) = 1 \oplus (0.0) = 1$$

$$B^2_7(1) = 0 \oplus (0.1) = 0$$

$$B^2_7(2) = 0 \oplus (1.1) = 1$$

$$B^2_7(3) = 1 \oplus (1.0) = 1$$

$$B^2_7(4) = 0 \oplus (0.1) = 0$$

$$B^2_7(5) = 0 \oplus (0.0) = 0$$

$$B^2_7(6) = 1 \oplus (1.0) = 1$$

$$B^2_7(7) = 0 \oplus (1.0) = 0$$

So, $B^2_7 = (1,0,1,1,0,0,1,0)$

If $p=7$ and $n=3$, then

$$B^3_7(i) = B^2_7(i) \oplus (M^3(i).B^2_7(i \oplus 4))$$

$$B^3_7(0) = 1 \oplus (0.0) = 1$$

$$B^3_7(1) = 0 \oplus (0.0) = 0$$

$$B^3_7(2) = 1 \oplus (0.1) = 1$$

$$B^3_7(3) = 1 \oplus (0.0) = 1$$

$$B^3_7(4) = 0 \oplus (1.1) = 1$$

$$B^3_7(5) = 0 \oplus (1.0) = 0$$

$$B^3_7(6) = 1 \oplus (1.1) = 0$$

$$B^3_7(7) = 0 \oplus (1.1) = 1$$

So, $B^3_7 = (1,0,1,1,1,0,0,1) = A_7$

Therefore,

$$B^3_0 = (1,0,1,0,1,1,0,1) = A_0$$

$$B^3_1 = (1,0,1,0,1,1,0,1) = A_1$$

$$B^3_2 = (0,0,1,0,1,0,1,1) = A_2$$

$$B^3_3 = (0,0,1,0,1,0,0,1) = A_3$$

$$B^3_4 = (1,1,0,1,0,1,1,1) = A_4$$

$$B^3_5 = (0,1,1,1,1,1,0,1) = A_5$$

$$B^3_6 = (1,0,0,1,1,0,1,1) = A_6$$

$$B^3_7 = (1,0,1,1,1,0,0,1) = A_7$$

According to definition-2, our RMC form must be formed by A_3 , due to its least number of "1". Now we must find out the actual polarity of input-variables of our minimal canonical forms. In this example, minimal vector of coefficients is A_3 . Therefore, $p=3$. The polarity of three input variables x_3, x_2, x_1 are found by converting this decimal number "3" into a binary number "011", i.e.,

$$\begin{array}{ccc} x_3 & x_2 & x_1 \\ 0 & 1 & 1 \end{array}$$

where "1" corresponding to a variable indicates the complemented form of that variable and "0" indicates the uncomplemented form of the variable. Finally, the minimal canonical

form of our Boolean function is:

$$f = \begin{bmatrix} 0,0,1,0,1,0,0,1 \end{bmatrix} \begin{bmatrix} x_3 & x_2 & x_1 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= 0 \cdot 1 \oplus 0 \cdot \bar{x}_1 \oplus 1 \cdot \bar{x}_2 \oplus 0 \cdot \bar{x}_2 \cdot \bar{x}_1 \oplus 1 \cdot x_3 \oplus 0 \cdot x_3 \cdot \bar{x}_1 \oplus 0 \cdot x_3 \cdot \bar{x}_2 \oplus 1 \cdot x_3 \cdot \bar{x}_2 \cdot \bar{x}_1$$

$$= \bar{x}_2 \oplus x_3 \oplus x_3 \cdot \bar{x}_2 \cdot \bar{x}_1$$

Proof:

	$x_2 x_1$	00	01	11	10
x_3	0	1	1		
	1	1		1	1

5.6.1 Algorithm:

Step 1: Set $p=0$.

step 2: Calculate $B_p^0 = V_p(i) = f(i \oplus p)$ for $0 \leq i \leq 2^n - 1$.

Step 3: Set $I=1$.

Step 4: Calculate the M^l vector.

step 5: Calculate

$$B^l_p(m) = B^{l-1}_p(m) \oplus (M^l(m) \cdot B^{l-1}_p(m \oplus 2^{l-1})) \quad \text{for } 0 \leq m \leq 2^n - 1$$

Step 6: $l=l+1$; if $l \leq n$ goto Step 4.

Step 7: Set $A_p = B^n_p$; $p = p + 1$; if $p < 2^n$ goto Step 2.

Step 8: Determine the A_p (vector of coefficients) for polarity coefficient p ($0 \leq p \leq 2^n$) which contains least number of '1'.

Step 9: Find the polarity of input-variables from the value of p for which A_p contains least number of '1'.

Step 10: Determine the minimal canonical form of the Boolean function.

CHAPTER 6

EXPERIMENTS AND RESULTS

6.1 Design of Experiments:

This thesis is an endeavor to analyze the performance of different logic synthesis and optimization techniques. Here our main purpose is to examine the time complexity, number of product terms and number of literals produced by these techniques. Due to the probabilistic nature of the computational time required for logic synthesis and optimization techniques, we have taken an elaborate process banked upon statistical experiments with different types of functions, instances of which are generated randomly. We consider random functions, partially symmetric functions, symmetric functions, positive functions, vacuous functions, unate functions, parity functions and Majority functions. Again we have taken different permutation of the same set of data. we have also calculate the number of product terms and number of literals produced by all of the above types of data. we have increased the number of variables to find the effect of time complexity of these techniques. Again for the same number of variables we have varied the number of minterms to find the time complexity.

To compare performances of various algorithms data set for random functions, partially symmetric functions, symmetric functions, positive functions, vacuous functions, unate functions, parity functions, majority functions were generated and number of products, number of literals and elapsed time have been taken as performance criteria. Random functions are generated by random() function of C language.

6.2 Quine-McCluskey Method:

To find the performance of Quine-McCluskey techniques, software has been developed in C. We have taken different types of functions and observed that for all the cases elapsed time increases if the adjacent minterms increases in the functions for the same number of variables.

When the number of minterms are around the same whereas the number of variables increases the elapsed time remains same as can be seen in output of (DS1, DS6-DS8, DS16; DS3-DS4, DS10-DS11, DS14, DS27, DS33, DS36, DS48-DS53).

When the minterms are not adjacent, then elapsed time increases with the increase of minterms as can be seen in output of(DS40-DS45).

For the partially symmetric and symmetric functions (DS22-DS30) time complexity is the same as in the case of randomly generated functions.

For positive, vacuous and unate functions (DS31-DS39) there are always adjacent minterms so in the SOP expression a minimized product terms results and the elapsed time depends on the degree of adjacency of the minterms.

For parity functions there are no adjacent minterms so final SOP expressions invariably contain minterms and here it is observed that time increases with the increasing number of minterms (DS40-DS45).

6.3 EXMIN2:

EXMIN2 Algorithm has been developed in C. At the start of the experiment several randomly generated functions up to 12 variables are applied and a number of samples are listed here. An attempt has been made to change the sequence of the operations in the algorithm. As a result we have to run the algorithm in slightly different fashion for the same set of data. These test are known as EXMIN2(Test1) and EXMIN2(Test2). In Test1 X-MERGE is applied after RESHAPE, DUAL-COMPLEMENT, X-EXPAND2 and X-EXPAND1 and in Test2 X-MERGE is applied after each of the above operations. In both cases X-MERGE is done repeatedly until there is no more reduction in number of products. It is observed that EXMIN2 and EXMIN2(Test2) give the same number of products and literals for almost all the data. For DS18 EXMIN2 gives fewer number of products and higher number of literals than EXMIN2(Test2) and for DS47 EXMIN2 gives equal number of products and fewer number of literals than EXMIN2(Test2). For most of the data EXMIN2 required less time than EXMIN2(Test2).

It is seen for the data (DS1-DS3, DS6, DS12-DS14, DS16, DS19-DS46) that EXMIN2 and EXMIN2(Test1) produce the same number of products and literals. For data (DS8-DS10, DS15) EXMIN2 produces equal number of products but higher number of literals. For data (DS11, DS17-DS18) EXMIN2 produces fewer number of products and literals than EXMIN2(Test1). For data (DS4) EXMIN2 produces fewer number of products and equal number of literals and for data (DS5) EXMIN2 produces equal number of products and fewer number of literals than EXMIN2(Test1). For data (DS47) EXMIN2(Test1) produces fewer number of products and higher number of literals. Table 6.1 shows that for data (DS54) EXMIN2(Test1) produce fewer number of products and literals than EXMIN2. In all these cases EXMIN2(Test1) required less time than EXMIN2.

For data (DS56), it is observed that before the step i of EXMIN2 algorithm the ESOP expression is as follows

$$BD \oplus bDEC \oplus Ae \oplus aBDE \oplus AbC \oplus 1 \oplus ACde$$

(Here capital letter means uncomplemented form and small letter means complemented form.)

i.e., number of products is 7 and the number of literals is 19. But in the final output number of products is 8 and number of literals is 24.

From the data (DS2, DS19, DS21, DS22, DS24) it is apparent that time is less when the degree of adjacency of minterms is high. For the same number of variables time required is increases with the increase of minterms (DS12, DS15). It is apparent from the data that there is no specific rules for time elapsed with the increase of minterms and variables. The elapsed time depends on the nature of data .

In most of the cases, it is apparent from the data that for the same number of variables time elapsed would be less if the resultant number of product is less as can be seen in outputs of (DS2-DS7). Again it is affected by the number of minterms.

For the same number of minterms, time increases with the increase of variables as can be seen in outputs of(DS48-DS53). For the different permutation of the same function it is seen that in some cases number of products is same but number of literals differs as can be seen in outputs of(DS54-DS58).

Table 6.10 shows number of products and literals produced by EXMIN2, EXMIN2(Test1), EXMIN2(Test2) for different permutation of the function of data (DS54). It is apparent from this table that in this case EXMIN2(Test1) gives better result.

6.4 Mlynarovic's Method for Minimization:

For the purpose of study the performance of Mlynarovic's method for minimization with different types of functions, software has been developed in C, similar to that of Quine-McCluskey method. To develop this software special care has been taken so that memory requirement would be less. This is achieved by bit wise operation of the variables. Here we applied the same set of data that were used in Quine McCluskey method. This method gives the fixed polarity EXOR expression. At the beginning by several randomly generated function, we have found that time required for 3 variables is 0.109890 seconds whatever the number of minterms (one to maximum). For 4 variables time required is almost 5 times of the time required for 3 variables. This relation is true for successive number of variables as can be seen in outputs of(DS1-DS5, DS12-DS18, DS21, DS47). It is observed that for a number of variables time elapsed is independent of the number of inputs for all kinds of function. For different permutation of the same function the output remains same as well as time required. For both odd and even parity function this method gives the minimal expression (DS40-DS45).

An attempt has been made to tailor Mlynarovic technique so that it produces minimum number of literals along with minimum number of products. To this end we examined the A_p vectors with minimum number of 1s. For a function there may be a number of A_p vectors with minimum number of 1s. It is observed that in such a case, number of products are the same but the number of literals may differ. For example, if we apply the following function

$$f = \Sigma (1, 2, 4, 5, 7, 8, 9, 10, 13, 14, 15)$$

in Mlynarovic's method then we have two solutions with different number of literals. These are

$$1 \oplus dC \oplus db \oplus Cb \oplus dCb \oplus dA \oplus dCbA$$

and

$$1 \oplus dC \oplus db \oplus Cb \oplus d \oplus da \oplus dCba$$

i.e., A_3 and A_{13} vectors contains the minimum number of 1s. It is seen that the first expression contains 15 literals and the second expression contains 13 literals.

6.5 Results:

In this section, number of products term, number of literals and the elapsed time for Quine-McCluskey, EXMIN2 and Mlynarovic methods are listed in tabular form for the different functions. The functions and the expressions obtained by applying different algorithms are listed in APPENDIX A. In the table

DS stands for Data Set.

DSn stands for Data Set number n, i.e., function number n.

The first, second, third and fourth columns contain respectively serial of data set, number of product terms, number of literals and the elapsed time. The second, third and fourth columns are split into three parts, first for results of Quine-McCluskey(QM), second for results of EXMIN2 algorithm(EX) and the third for results of Mlynarovic(ML). Table 6.1 contains results of applying random functions on various algorithms. Similarly Tables 6.2-6.8 corresponds to partially symmetric, symmetric, positive, vacuous, unate, parity and majority functions. Table 6.9 contains random functions with higher number of variables (DS47, DS59), functions with the same number of minterms but different number of variables (DS48-DS53) and different permutations of the same function.

Table 6.1 Results of Statistical Experiments with Random Functions:

DS	No of Product			No of Literals			Elapsed Time (in seconds)		
	QM	EX	ML	QM	EX	ML	QM	EX	ML
1	3	3	3	5	4	4	.054945	.494505	.109890
2	2	2	2	4	2	2	.054945	.494505	.549451
3	5	4	7	12	11	15	.109890	.549451	.549451
4	6	5	6	16	12	12	.109890	.659341	.549451
5	8	6	8	27	17	25	.164835	.769231	2.692308
6	4	4	10	17	17	33	.054945	.494505	2.747253
7	3	3	5	12	12	19	.054945	.549451	2.747253
8	4	4	8	16	16	24	.054945	.769231	2.747253
9	6	6	13	28	26	39	.109890	.714286	2.692308
10	9	7	12	40	28	31	.109890	.824176	2.692308
11	5	5	9	20	20	27	.109890	.549451	2.692308
12	6	6	14	35	33	55	.109890	.769231	13.351648
13	8	8	20	39	34	78	.164835	1.043956	13.296703
14	9	8	20	52	38	73	.109890	1.098901	13.296703
15	18	13	26	88	53	82	.384615	2.637363	13.186813
16	4	4	16	26	26	77	.054945	.439560	63.571429
17	24	18	49	139	90	178	.604396	7.307692	62.527473
18	28	21	55	149	95	203	1.318681	10.494505	62.307692
19	1	1	1	1	1	1	.769231	.109890	13.351648
20	2	2	2	4	2	2	.494505	.549451	13.296703
21	1	1	1	1	1	1	35.989011	.6549341	296.04396

DSn = Data Set number n, QM = Quine-McCluskey

EX = EXMIN2, ML = Mlynarovic

Table 6.2 Results of Statistical Experiments with Partially Symmetric Functions:

DS	No of Product			No of Literals			Elapsed Time (in seconds)		
	QM	EX	ML	QM	EX	ML	QM	EX	ML
22	2	2	2	4	2	2	.054945	.439560	.549451
23	2	3	3	4	8	8	.054945	.439560	.549451
24	2	2	2	4	2	2	.054945	.439560	.549451

Table 6.3 Results of Statistical Experiments with Symmetric Functions:

DS	No of Product			No of Literals			Elapsed Time (in seconds)		
	QM	EX	ML	QM	EX	ML	QM	EX	ML
25	5	5	6	25	20	25	.054945	.439560	2.747253
26	6	6	6	36	30	30	.054945	.439560	13.351648
27	10	9	10	50	31	30	.109890	1.483516	2.692308
28	11	8	14	44	25	30	.164835	1.043956	2.692308
29	10	8	10	40	20	20	.274725	1.098901	2.692308
30	11	8	14	44	25	30	.219780	.934066	2.692308

Table 6.4 Results of Statistical Experiments with Positive Functions:

DS	No of Product			No of Literals			Elapsed Time (in seconds)		
	QM	EX	ML	QM	EX	ML	QM	EX	ML
31	2	2	2	4	4	4	.054945	.439560	.549451
32	2	2	2	6	4	4	.054945	.439560	.549451
33	3	3	3	9	8	8	.109890	.549451	2.747253

Table 6.5 Results of Statistical Experiments with Vacuous Functions:

DS	No of Product			No of Literals			Elapsed Time (in seconds)		
	QM	EX	ML	QM	EX	ML	QM	EX	ML
34	1	1	1	2	2	2	.054945	.054945	2.747253
35	4	4	4	16	8	8	.109890	.604396	2.747253
36	3	3	3	8	7	7	.109890	.549451	2.747253

Table 6.6 Results of Statistical Experiments with Unate Functions :

DS	No of Product			No of Literals			Elapsed Time (in seconds)		
	QM	EX	ML	QM	EX	ML	QM	EX	ML
37	2	2	3	5	6	9	.054945	.439560	.549451
38	2	2	3	6	7	11	.054945	.384615	2.747253
39	3	3	5	6	9	13	.274725	.494505	2.747253

Table 6.7 Results of Statistical Experiments with Parity Functions:

DS	No of Product			No of Literals			Elapsed Time (in seconds)		
	QM	EX	ML	QM	EX	ML	QM	EX	ML
40	16	5	5	80	5	5	.164835	.769231	2.692308
41	32	6	6	192	6	6	.494505	1.318681	13.241758
42	64	7	7	448	7	7	1.153846	4.065934	62.857143
43	16	5	5	80	5	5	.219780	.714286	2.692308
44	32	6	6	192	6	6	.494505	1.318681	13.241758
45	64	7	7	448	7	7	1.153846	4.010989	62.802198

Table 6.8 Results of Statistical Experiments with Majority Function:

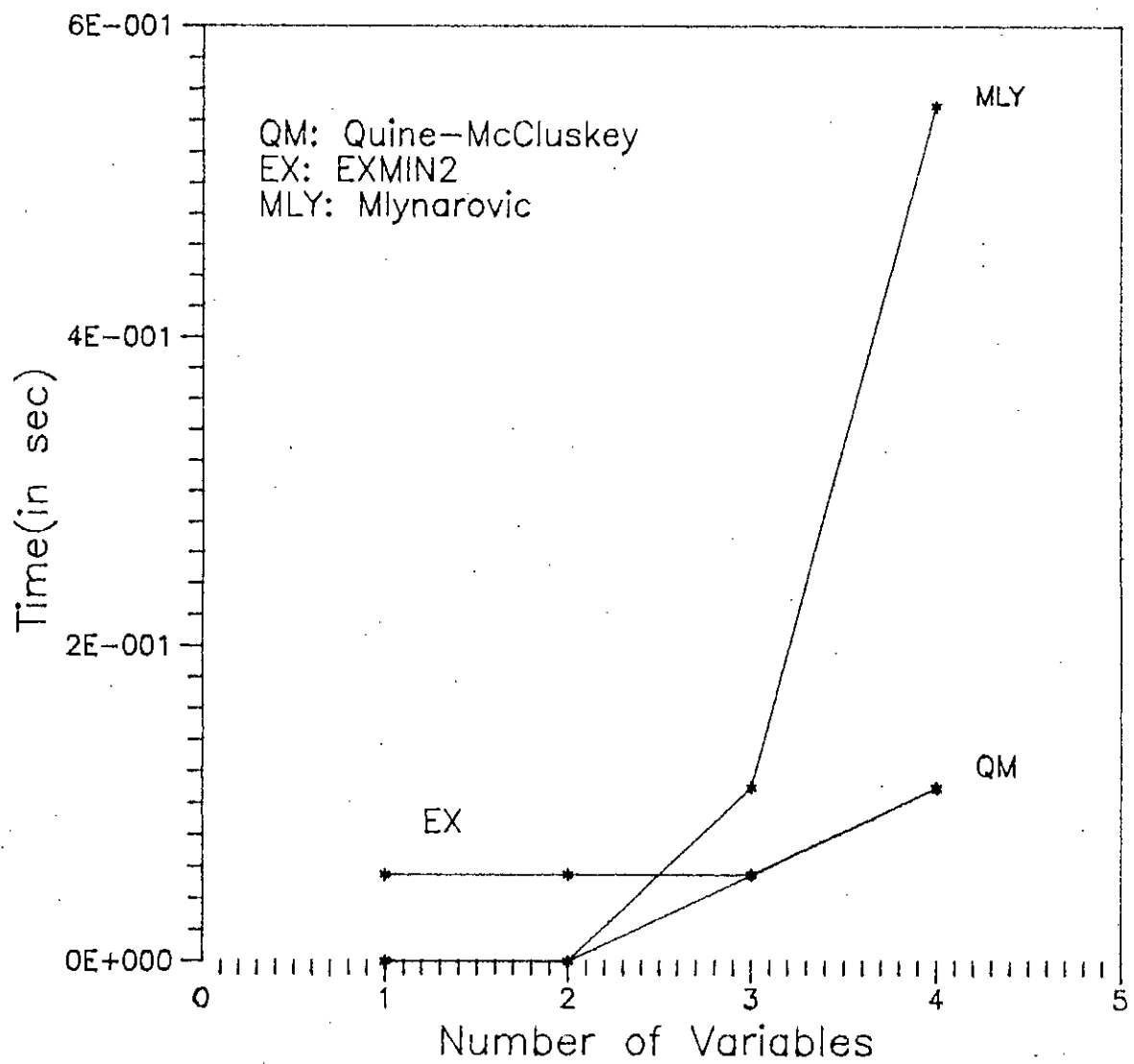
DS	No of Product			No of Literals			Elapsed Time (in seconds)		
	QM	EX	ML	QM	EX	ML	QM	EX	ML
46	10	8	15	30	27	50	.164835	1.043956	2.692308

Table 6.9 Results of Statistical Experiments with Random Functions (Miscellaneous):

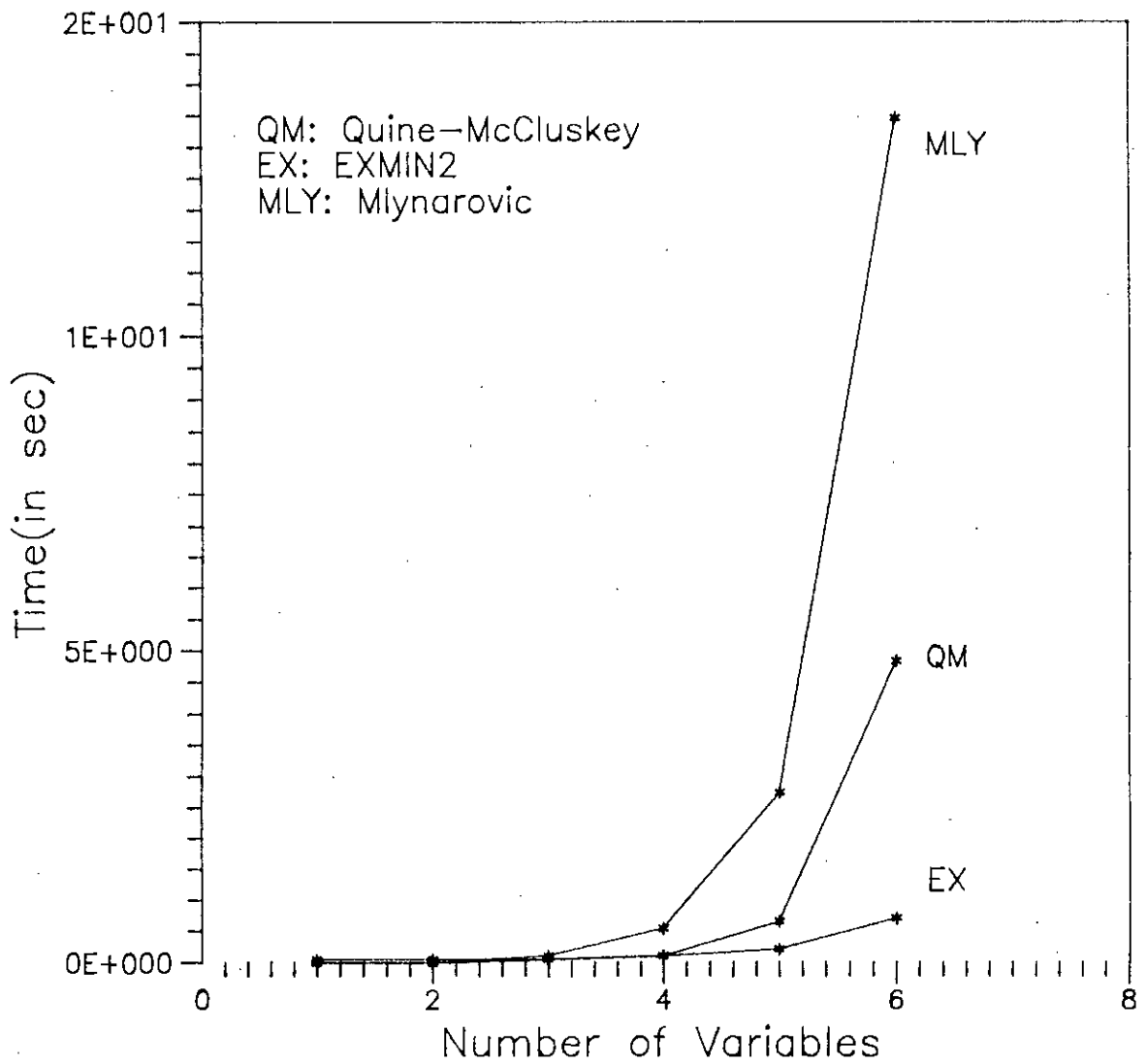
DS	No of Product			No of Literals			Elapsed Time (in seconds)		
	QM	EX	ML	QM	EX	ML	QM	EX	ML
47	56	50	125	417	313	645	5.604396	88.076923	1338.4615
48	4	4		11	12		.054945	.494505	
49	5	5		20	19		.054945	.714286	
50	5	5		26	25		.054945	.659341	
51	8	7		56	43		.109890	1.098901	
52	8	7		64	52		.109890	.934066	
53	7	7		62	58		.109890	1.373626	
54		8			25			1.153846	
55		8			24			1.208791	
56		8			24			1.098901	
57		8			21			.824176	
58		8			24			1.318681	
59	137			1782			18.681319		

Table 6.10 Comparison Among EXMIN2, EXMIN2(Test1), EXMIN2(Test2) for Different Permutation of a Function:

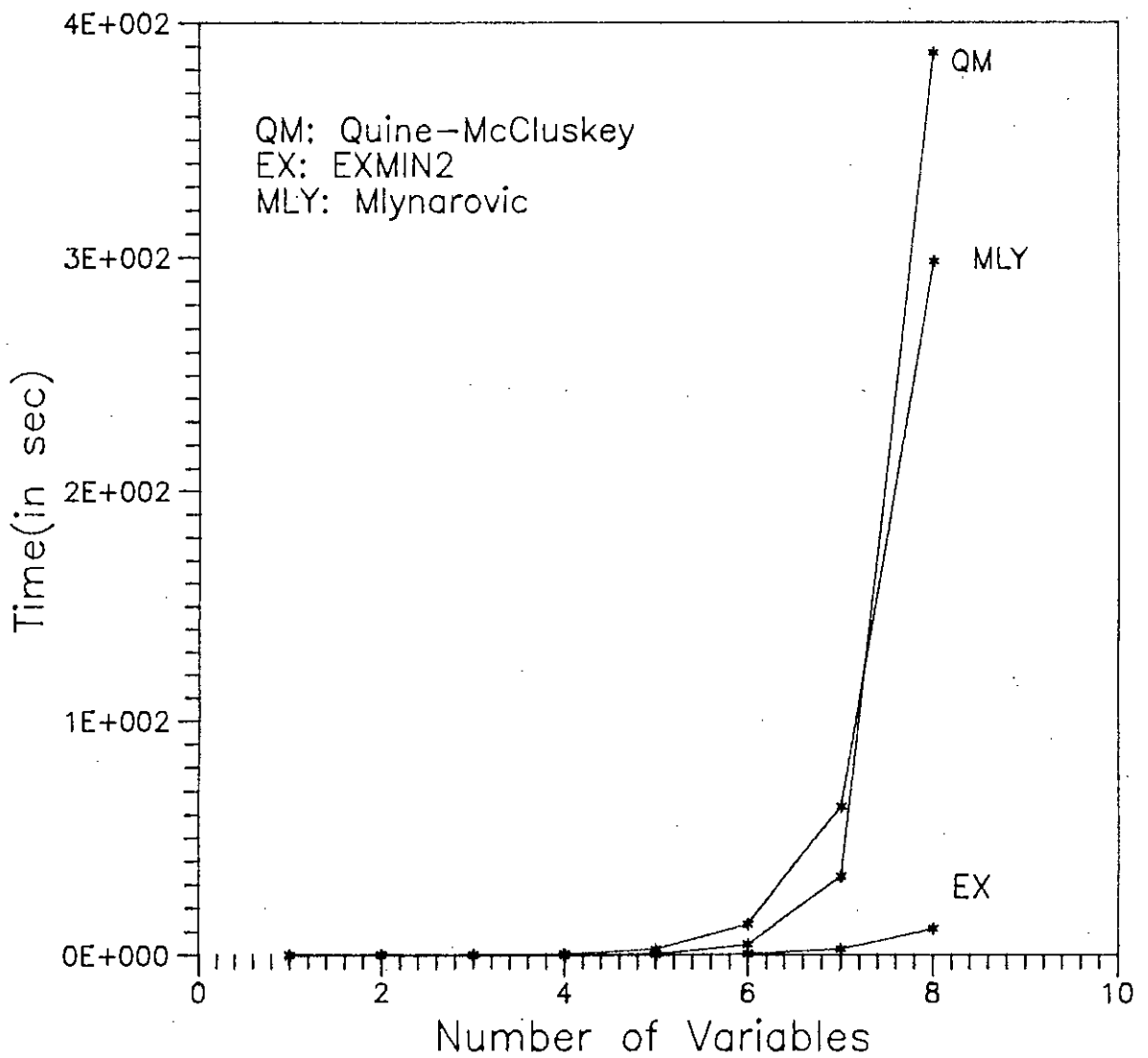
DS	EXMIN2		EXMIN2(Test1)		EXMIN2(Test2)	
	No. of Products	No. of Literals	No. of Products	No. of Literals	No. of Products	No. of Literals
54	8	25	7	22	8	25
55	8	24	7	20	8	24
56	8	24	8	24	7	24
57	8	21	7	20	7	22
58	8	24	7	20	8	25



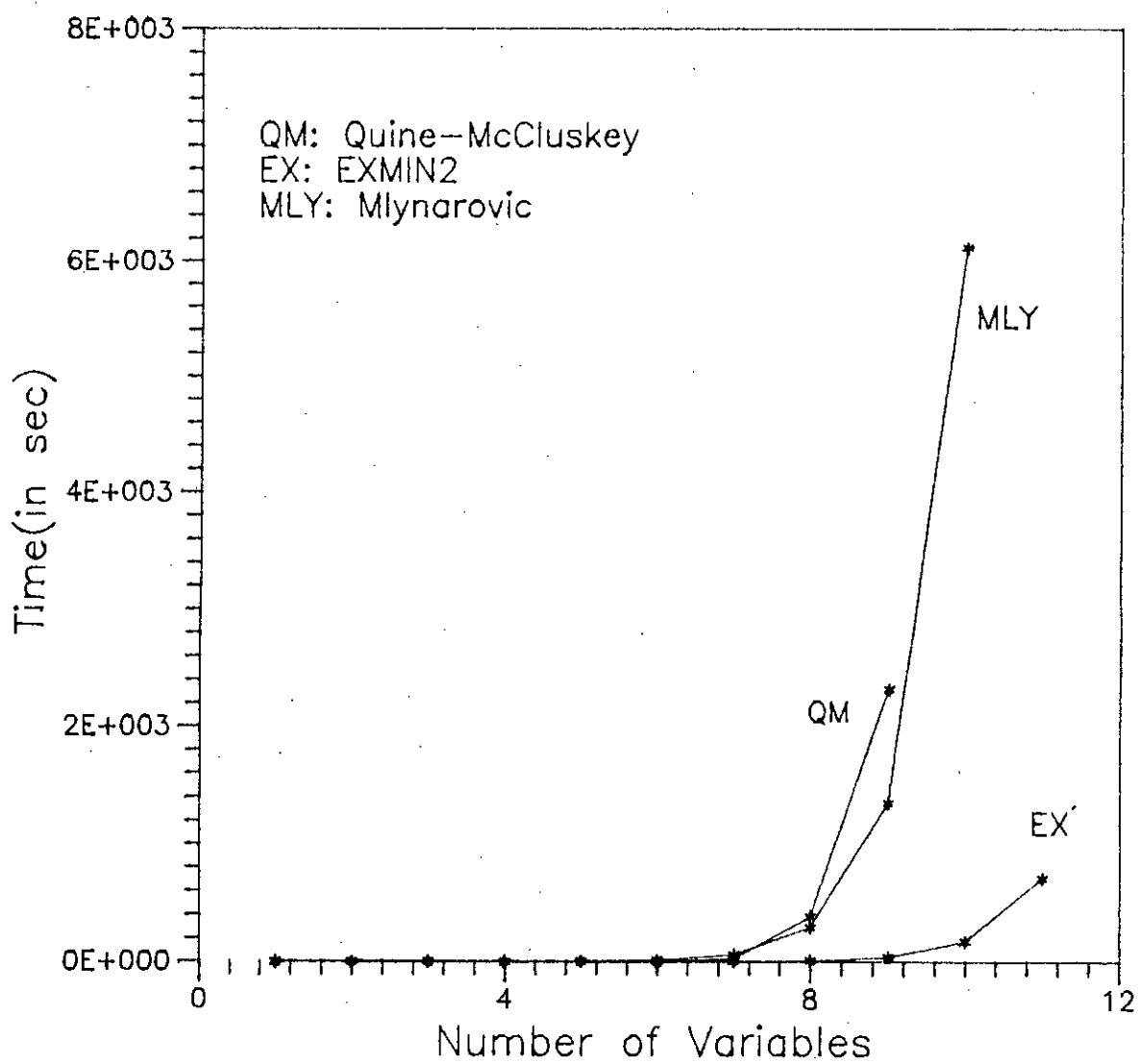
Graph 6.1 Comparison of Worst Case Elapsed Time



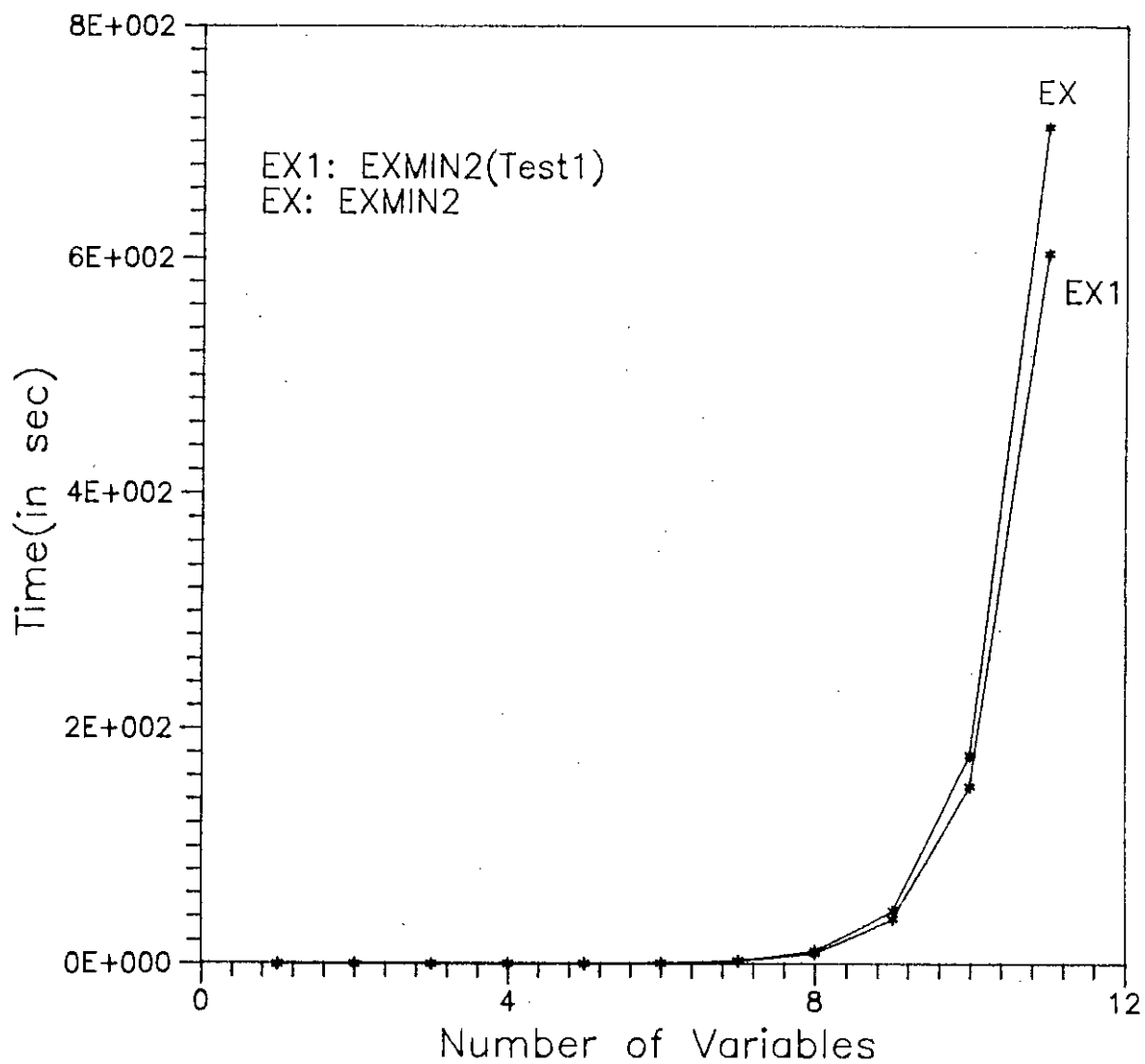
Graph 6.2 Comparison of Worst Case Elapsed Time



Graph 6.3 Comparison of Worst Case Elapsed Time



Graph 6.4 Comparison of Worst Case Elapsed Time



Graph 6.5 Comparison of Worst Case Elapsed Time between EXMIN2 and EXMIN2(Test1)

6.6 Comparison Among the Minimization Techniques:

All the data except a few show that time required for Mlynarovic is the highest and for the Quine-McCluskey it is lowest. It is apparent from the data (DS19, DS21) that when the decimal representation of minterms are consecutive even or odd numbers then Quine-McCluskey required more time than the EXMIN2. It is true for functions which have non-overlapping loops. However in all these cases time elapsed in Mlynarovic is the highest as can be seen in outputs of(DS19-DS21).

EXMIN2 gives the fewer number of products and literals than Mlynarovic techniques for all kinds of inputs except for parity function, vacuous function, positive function. For these functions both give the same number of products and literals as can be seen in outputs of(DS31-DS36, DS40-DS45).

Most of the cases EXMIN2 gives the fewer products than Quine-McCluskey. In some cases number of products are same but literals are greater than Quine-McCluskey as can be seen in outputs of(DS1, DS9, DS12, DS13, DS20). For these cases nature of the data cannot be determined. From data (DS25-DS30) it is seen that for symmetric function number of products are equal or fewer than Quine-McCluskey and literals are always fewer. For parity function EXMIN2 gives the fewer products and literals than Quine-McCluskey as can be seen in outputs of(DS40-DS45).

For Mlynarovic and Quine-McCluskey method there is not any specific relationships between them in the number of products and number of literals produced. For a specific data it is observed that Mlynarovic technique is better than Quine-McCluskey and vice versa. For unate function(DS37-DS39) Mlynarovic technique gives higher number of products and literals than Quine-McCluskey method. On the other hand for parity function(DS40-DS45) Mlynarovic technique gives smaller number of products and literals than Quine-McCluskey method.

CHAPTER 7

CONCLUSIONS AND RECOMMENDATIONS

7.1 Conclusions:

From the results given in chapter six, it is obvious that performances of various minimization techniques are sensitive to the nature of the data and adjacency of the minterms. For Quine-McCluskey data (DS1-DS21) show that the response time increases with the increase of the adjacent minterms, because in this case higher cubes are generated for the higher degree of adjacency. For non-adjacent minterms or for the function with a lower degree of adjacency the time elapsed is the same for the same number of minterms independent of number of variables as can be seen in outputs of (DS48-DS53), because higher cubes need not be generated. When the minterms are not adjacent, as in the case of data (DS40-DS45), the elapsed time increases with the increase of minterms because overhead required for searching and sorting increases with the increase of input data.

For Quine-McCluskey, graph 6.4 and data (DS59) show that worst case response time for 9 variables is much higher than 14 variable randomly generated function. It is apparent that Quine-McCluskey may be applied to any number of variables but its response time increases with the increase of minterms and it is maximum in the worst case. Actually, with the increase of minterms, overhead required for the additional formations of linked-list and searching increases and becomes maximum in the worst case for a given number of variables.

It is seen that for parity function Quine-McCluskey produces the minterms, i.e., there is

no minimization. So to minimize the hardware complexity for this function we have to go in the Exclusive-OR domain. It is apparent from the Quine-McCluskey method that different permutations of the same function do not affect the number of products and number of literals, because all possible cubes are generated here.

EXMIN2 gives the products in mixed polarity and its elapsed time decreases with the increase of the degree of adjacency. In EXMIN2 degree of adjacency is not similar to that of Quine-McCluskey because here 'zero' entry may be an adjacent cell if it is in even number of loops. So a number of solutions may be possible in EXMIN2 if we permute the input function even if the number of products and number of literals are the same.

EXMIN2 is formed with a number of rules. The first objective of this algorithm is to minimize the number of products and the second objective is to minimize the number of literals. As a consequence, rules responsible for minimizing the product term commence first in the algorithm. Since minimization of product terms and minimization of literals are done by different rules, so minimization of product terms and literals are not achieved simultaneously for permutation of a given function as can be seen in outputs of (DS54-DS58).

Graphs 6.1-6.5 and data (44,78) show the variation of response time with the increase of adjacent minterms for EXMIN2. It is apparent that a function with adjacent minterms required less time than a function with a higher number of nonadjacent minterms. Thus it can be concluded that when EXMIN2 deals with a function with a greater number of nonadjacent minterms, response time would be higher due to the inclusion of zero terms in K-Map.

It is seen that for parity function EXMIN2 gives a fewer number of products and literals than Quine-McCluskey. So to minimize such functions, ESOP domain is appropriate.

In step(c) of EXMIN2 algorithm, the sequence of operations has been modified. As a result EXMIN2(Test1) and EXMIN2(Test2) do not produce any significant results. From Table 6.10 it is observed that for function DS54 and its different permutations, EXMIN2(Test1) gives fewer number of products and literals. So it is not clear whether best sequence of operations in EXMIN2 is unique or not. Such variations of EXMIN2 have been done in different ways without fruitful results.

From the data (DS54-DS58), it is apparent that the output of EXMIN2 is not unique. With different permutations of inputs it is seen that number of products is the same but number of literals differs. So we conclude that we can get a minimal expression among the different solutions of EXMIN2 for a given function. It can also be seen from graph 6.5 that response time of EXMIN2 is higher than the EXMIN2(Test1) in the worst case and it is true for any kind of input data.

It is observed from data DS56 and Art. 6.3 that the process of minimization in EXMIN2 is not always convergent. To avoid this difficulty we have to include a step between step h and step i of EXMIN2. In this step we have to store the minimum expression before further process, i.e., SPLIT operation. The final output would be compared with this expression and the minimum expression would be the final output.

Graphs 6.1-6.4 and all the data show that response time of Mlynarovic's method is insensitive to the number of minterms and the adjacency of minterms for a given number of variables. It is observed from all the data that response time for a 5 variable function is a constant times the time required for a 4 variable function and this constant relation is true for any consecutive number of variables. It is observed that even if the number of minterms is one, then time elapsed for that function is the same as the worst case for a given number of variables. This is due to the fact that whatever the number of minterms be, the algorithm generates all the vectors. As a result overhead is the same.

Data(DS40-DS45) show that Mlynarovic's method gives the minimum solution for the parity function. It is interesting to note that only for this kind of functions both EXMIN2 and Mlynarovic's method give the same number of products and literals though the exclusive-OR sum of products expressions are different.

In Mlynarovic method it is seen that there may be a number of minimum A_p vectors for some functions but number of literals in the expressions may differ. In this case we have to choose the A_p vector which produces minimum number of literals. As a result Mlynarovic technique should be modified to overcome this problem. One solution of this problem is by software implementation, where we have to generate all the expressions for A_p vectors with minimum number of 1s, then choose the expression with minimum number of literals.

Graphs 6.1-6.4 shows the comparison of Quine-McCluskey, EXMIN2 and Mlynarovic's techniques for different number of variables in the worst case(i.e., all possible minterms are present for a given number of variable). For small number of variables response time in the Mlynarovic's technique is always higher than Quine-McCluskey, but when number of variables is higher than 7, response time for the Quine-McCluskey technique is higher than Mlynarovic's technique. In all these cases response time for EXMIN2 is the lowest. For the average number of minterms response time for the EXMIN2 and Quine-McCluskey depends on the nature of the data. If the minterms in a function are adjacent in nature in the SOP domain then the response time for the Quine-McCluskey algorithm is greater than EXMIN2. However, in all these cases the response time for Mlynarovic's method is the maximum.

EXMIN2 gives the fewer number of products and literals than Mlynarovic's method for all kinds of data. Most of the cases EXMIN2 gives fewer products than Quine-McCluskey method but in the case of symmetric functions and parity functions (DS25-DS30, DS40-DS45) it gives fewer number of products and literals. This is true for positive functions

(DS31-DS33) and vacuous functions (DS34-DS36). Data (DS37-DS39) show that in the case of unate function Quine-McCluskey gives the fewer number of products and literals than EXMIN2.

7.2 Suggestions for Further Study:

The Mlynarovic method, time elapsed is independent of number of minterms. We can also get Fixed Polarity Reed-Muller Expression by applying equations 3.1 and 3.2. In this case, we have 2^n FPRMEs and among them we have to choose the FPRME with minimum number of products. The time complexity of this technique would be the same as Mlynarovic but in this case elapsed time would be dependent on the number of minterms.

In Mlynarovic method, there may be multiple solutions but in these solutions number of literals may not be same. So one can attempt to solve this problem.

To get a mixed polarity exclusive-OR sum of products expression, we have simulated EXMIN2 algorithm. It is seen that in this algorithm, the process of minimizing the products is not always convergent. Step i of EXMIN2 algorithm applies the variable x_i in SPLIT operation which is responsible for increasing the minimum number of the products in $\bar{x}_i \cdot F_i \oplus x_i \cdot F_i$. Therefore, one can attempt to examine this step for different variables and study their number of products and literals.

Transformation from SOP domain to ESOP domain (Art. 5.4.4), i.e., HEALEX procedure may be applied to get a mixed polarity expression. In this procedure, determination of disjoint cube in RM domain is a vital one. Due to unavailability of literature this technique has not been tested in this study. Research can be carried out in this direction.

APPENDIX A

DATA AND SOP/ESOP/FPRME EXPRESSIONS

This appendix contains a sample of the data and the corresponding expressions produced by the different algorithms. Here,

symbol # stands for EXOR operator.

CAPITAL LETTER stands for uncomplemented variable.

SMALL LETTER stands for complemented variable.

NEGATIVE NUMBER stands for termination of inputs.

DS_n stands for data set number n.

FPRME stands for fixed polarity RME produced by Mlynarovic method.

DS1. Minterms are : 0, 1, 3, 5, 6, 7, -1

SOP Expression is: $C + ab + AB$

ESOP Expression is: $1 \# cA \# cB$

FPRME is: $1 \# cB \# cA$

DS2. Minterms are: 0, 2, 5, 7, 8, 10, 13, 15, -1

SOP Expression is: $bd + BD$

ESOP Expression is: $b \# D$

FPRME is: $d \# B$

DS3. Minterms are: 0, 2, 3, 4, 6, 8, 11, 12, 13, 14, -1

SOP Expression is: $cd + Bd + bCD + ABc + ad$

ESOP Expression is: $bC \# bCad \# d \# ABDc$

FPRME is: $1 \# D \# DCb \# DA \# DCA \# DbA \# CbA$

DS4. Minterms are: 1, 2, 3, 4, 6, 7, 8, 11, 12, 13, 14, -1

SOP Expression is: $aC + Bd + abD + bCD + Acd + ABc$

ESOP Expression is: $1 \# acBD \# bcd \# DAC \# bA$

FPRME is: $1 \# Dc \# cb \# DA \# bA \# DcbA$

DS5. Minterms are: 0, 2, 4, 5, 6, 8, 10, 12, 14, 15, 16, 18, 21, 23, 25, 26, 28, 29, 31, -1

SOP Expression is: $ac + bce + cDe + ACE + ABdE + abCd + BCde + aBCD$

ESOP Expression is: $eCa \# E \# BAd \# CEdB \# ECDba \# c$

FPRME is: $1 \# E \# EdC \# ECB \# CA \# EdCA \# dBA \# ECBA \# EdCBA$

DS6. Minterms are: 20, 21, 2, 13, 3, 19, -1

SOP Expression is: $AbCd + abcD + bcDE + aBCdE$

ESOP Expression is: $bcD \# aBCdE \# bcDeA \# AbCd$

FPRME is: $EDb \# Cb \# DCb \# EDCb \# ECa \# EDCa \# Dba \# EDba \# Cba \# ECba$

DS7. Minterms are: 27, 11, 31, 3, 29, -1

SOP Expression is: $acDE + ABCE + BcDE$

ESOP Expression is: $cDEAb \# EDc \# EBAC$

FPRME is: $EDc \# EDcA \# EBA \# EcBA \# EDcBA$

DS8. Minterms are: 5, 9, 21, 20, 29, 1, 8, -1

SOP Expression is: $aBcd + AbCd + ACdE + abdE$

ESOP Expression is: $CdA \# adBc \# adbE \# CdAeB$

FPRME is: $Ed \# EdB \# dcB \# dA \# EdA \# dcA \# dBA \# EdcBA$

DS9. Minterms are: 23, 13, 17, 28, 27, 30, 11, 2, -1

SOP Expression is: $ABCe + BcDE + AbCDE + aBCdE + AbcdE + abcDe$

ESOP Expression is: $EABd \# abcDe \# ACeB \# cBED \# EdBaC \# Acbe$

FPRME is: $ED \# EC \# Db \# ECb \# DCb \# EDCb \# CA \# EDCA \# EbA \# DbA \# CbA \# ECbA \# DCbA$

DS10. Minterms are: 12, 19, 18, 25, 5, 2, 29, 30, 0, 24, 9, 15, -1,

SOP Expression is: $AbcD + ABdE + ABcd + BcdE + abce + aBCde + abCdE + ABCDE + aBCDE$

ESOP Expression is: $cbDA \# aEbdC \# aebc \# aCB \# DBeC \# dEB \# BceAd$

FPRME is: $D \# DC \# B \# eCB \# DCB \# ea \# Da \# Ca \# eDCa \# DBa \# eDBa \# eCBa$

DS11. Minterms are: 13, 0, 19, 16, 1, 10, 4, 7, 15, 5, -1,

SOP Expression is: $aCE + abd + bcde + AbcDE + aBcDe$

ESOP Expression is: $CaE \# Cabde \# AbcE \# bdc \# aBcDe$

FPRME is: $Ecb \# dcb \# Ea \# ca \# dca \# Edca \# dba \# Edba \# cba$

DS12. Minterms are: 40, 38, 1, 16, 51, 57, 9, -1

SOP Expression is: $abdeF + AbCdef + AbcDEf + aBcdef + ABcdEF + ABCdeF$

ESOP Expression is: $aBcdef \# deFba \# AdCeB \# AdCef \# AdBFcE \# AbcDEF$

FPRME is: $ed \# fed \# edB \# fedB \# fedcB \# edA \# fcA \# fecA \# fdcA \# fedBA \# fcBA \# fecBA \# dcBA \# fedcBA$

DS13. Minterms are: 45, 19, 57, 40, 55, 63, 30, 47, 62, 43, 58, 26, 52, 61, 54, 35, -1,

SOP Expression is: $ACDF + ABDE + BCEf + ABCeF + AbdEF + ABcDf + aBcdEF + AbCdef$

ESOP Expression is: $BEC \# BEF \# BEcFaD \# ACF \# ACdb \# ACdbfE \# AEFd \# ABcDf$

FPRME is: $FEdB \# ECB \# FECB \# FEdCB \# FEaA \# FCA \# dCA \# EdCA \# FEdCA \# BA \# FBA \# FEBA \# dBA \# FdBA \# FEdBA \# CBA \# FCBA \# FECBA \# FdCBA \# EdCBA$

DS14. Minterms are: 36, 51, 16, 2, 44, 26, 39, 52, 1, 63, -1

SOP Expression is: $AbDef + AcDef + ABcdEF + aBcdef + abcdEf + aBCdEf + AbcDEF + abcdeF + ABCDEF$

ESOP is : $AEFBC \# EdBA \# acdf \# adceb \# ADef \# ADefBC \# EdBf \# AEFcbD$

FPRME is: $fEd \# fdc \# fEdb \# dcb \# EdcB \# EA \# fEA \# EdA \# fcA \# EcA \# fEdcA \# fba \# Eba \# fdbA \# EdbA \# fEdbA \# fcbA \# fEcbA \# dcbA \# fdcba$

DS15. Minterms are: 35, 60, 55, 5, 24, 23, 29, 58, 53, 34, 18, 57, 48, 26, 1, 47, 54, 2, 19, 6, 12, 50, 7, 15, 10, 41, 17, 33, 44, 8, -1

SOP Expression is: $aCdf + BdEf + ACDef + ABcDF + ACdeF + ABcdf + bCDEF + aBCDeF + AbcdE + abceF + aBcEF + abcEf + cdEf + AbcdF + ABcDE + abcDF + acdeF + bCDef$

ESOP Expression is: $DcfbaE \# bDAcF \# bCA d \# Fb \# dbE \# eC \# dbEFA \# dfB \# BcEAD \# dfaceB \# caFB \# FeDB \# eCaB$

FPRME is: $f \# ED \# c \# Ec \# fDc \# DB \# fEDB \# EcB \# fEcB \# fEDcB \# A \# EA \# fEA \# DA \# EDA \# fEDA \# cA \# fEBA \# DBA \# EDBA \# fEDBA \# cBA \# EcBA \# fEcBA \# fDcBA \# EDcBA$

DS16. Minterms are: 23, 37, 46, 33, 55, 94, -1

SOP Expression is: $aCdeFG + aBcdfG + aBcDEFG + AbCDEFG$

ESOP Expression is: $aCdeFG \# AbCDEFG \# aBcdfG \# aBcDEFG$

FPRME is: FEDC # GFEDC # FEDCB # GFEDCB # GFECa # FEDCa # GBa # GFBa
GDBa # GFDBa # FEDBa # GFEDBa # GCBa # GFCBa # GDCBa # GFDCBa

DS17. Minterms are: 41, 66, 49, 72, 7, 97, 86, 75, 4, 12, 77, 92, 105, 121, 30, 99, 26,
101, 46, 36, 104, 13, 60, 11, 42, 10, 126, 8, 93, 79, 114, 58, 39, 30, 81, 68, 29, 64, 113,
62, 40, 54, 24, -1

SOP Expression is: BcDef + ABefG + bDEfG + aCDFg + abDeg + aBDFg + Abcdeg
+ BCdefG + acdEFG + ABcdeG + ABcdfG + acdEfg + AbCDEf + aBCDEg + BCDEFg
+ ACdefG + aBCEfg + AbCdEfg + ABCdeFg + bcDeFG + cDefg + AbcDFG + abcEfg
+ bcdEfg

ESOP Expression is: DcabE # Dfe # GcDb # gcd # cgaF # acfgde # DcabEfg # FbAEd
BAcd # dEFcG # FbAEdGC # DgaC # CeGfBa # ACgFB # CeGfAd # DCfb #
eADCgB # EFgdCB

FPRME is: ge # fe # gD # gfd # fc # gfc # ec # gfec # gDc # fDc # eDc # geDc # gfeDc
gb # gfb # fDb # geDb # gcb # fcb # gfcB # ecb # fecb # Dcb # gDcb # eDcb # geDcb
feDcb # ga # gfa # gDa # geDa # feDa # gfeDa # ca # gfeca # Dca # gDca # geDca
feDca # geba # feba # gDba # gfdBa # gfeDba # fcba # ecba # gecba # Dcba # gDcba

DS18. Minterms are: 0, 2, 4, 5, 7, 8, 10, 13, 16, 17, 19, 21, 23, 25, 27, 28, 29, 31, 34,
36, 37, 39, 42, 47, 48, 54, 55, 57, 58, 59, 63, 67, 69, 71, 72, 74, 75, 78, 81, 83, 85, 86,
88, 90, 92, 95, 96, 97, 100, 101, 104, 106, 109, 112, 113, 115, 117, 118, 120, 125, 127,
-1

SOP Expression is: aceFg + abEfg + aCDeG + CDEFG + aBEFG + AbDeg + ACdeG
+ ABEfG + AbcDFg + ACdEfg + ABefg + abcdfg + abCdef + bCDEfg + AbcdFG +
abCG + abceg + acdEf + abdEG + bcdEG + bdEFG + cDeFg + BcdEf + ABcdf + aCdefg
+ aBDeFg + aBCdEF + AbceFG

ESOP Expression is: fBDE # ace # fBc # aGcF # beFGc # GdEbc # DgAc # adbf #
DgAEB # DgAcfE # aGcb # fBcAD # aDC # fBcG # AdCG # adbfec # abCdeFG #
abCgE # aFGECD # DCbg # ECF

FPRME is: GF # Fe # d # Fd # ed # Fed # GeC # dC # GFdC # GFedC # Gb # eb #
Feb # GFeb # Gdb # GedB # Cb # GCb # FCb # FeCb # dCb # edCb # GedCb # FedCb
GFedCb # GA # eA # FeA # GFeA # GdA # GFdA # FedA # GFedA # GFCA #
GeCA # FeCA # GdCA # edCA # GedCA # FedCA # FbA # ebA # GebA # dbA #
GdbA # GedbA # FedbA # FCbA # eCbA # GeCbA # dCbA # FdCbA # edCbA #
FedCbA # GFedCbA

DS19. Minterms are: 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, -1

SOP Expression is: f

ESOP Expression is: f

FPRME is: f

DS20. Minterms are: 0, 2, 5, 7, 8, 10, 13, 15, 16, 18, 21, 23, 24, 26, 29, 31, 32, 34, 37, 39, 40, 42, 45, 47, 48, 50, 53, 55, 56, 58, 61, 63, -1

SOP Expression is: df + DF

ESOP Expression is: d # F

FPRME is: f # D

DS21. Minterms are: 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146, 148, 150, 152, 154, 156, 158, 160, 162, 164, 166, 168, 170, 172, 174, 176, 178, 180, 182, 184, 186, 188, 190, 192, 194, 196, 198, 200, 202, 204, 206, 208, 210, 212, 214, 216, 218, 220, 222, 224, 226, 228, 230, 232, 234, 236, 238, 240, 242, 244, 246, 248, 250, 252, 254, -1

SOP Expression is: h

ESOP Expression is: h

FPRME is: h

Partially Symmetric Function(22-24):

DS22. Minterms are: 4, 5, 6, 7, 8, 9, 10, 11, -1

SOP Expression is: aB + Ab

ESOP Expression is: a # b

FPRME is: B # A

DS23. Minterms are: 3, 7, 11, 12, 13, 14, 15, -1

SOP Expression is: CD + AB

ESOP Expression is: AB # ABCD # CD

FPRME is: DC # BA # DCBA

DS24. Minterms are: 2, 3, 6, 7, 8, 9, 12, 13, -1

SOP Expression is: $aC + Ac$

ESOP Expression is: $a \# c$

FPRME is: $C \# A$

Symmetric Function(25-30):

DS25. Minterms are: 15, 23, 27, 29, 30, -1

SOP Expression is: $aBCDE + AbCDE + ABcDE + ABCdE + ABCDe$

ESOP Expression is: $EBDC \# CABde \# ABDE \# BCA \# CAED$

FPRME is: $EDCB \# EDCA \# EDBA \# ECBA \# DCBA \# EDCBA$

DS26. Minterms are: 31, 47, 55, 59, 61, 62, -1

SOP Expression is: $aBCDEF + AbCDEF + ABcDEF + ABCdEF + ABCDeF + ABCDEF$

ESOP Expression is: $ADECf \# ADECb \# AFCBe \# AFCBd \# AEFBD \# FBDEC$

FPRME is: $FEDCB \# FEDCA \# FEDBA \# FECBA \# FDCBA \# EDCBA$

DS27. Minterms are: 3, 5, 6, 9, 10, 12, 17, 18, 20, 24, -1

SOP Expression is: $abcDE + abCdE + abCDe + aBcdE + aBcDe + aBCde + AbcdE + AbcDe + AbCde + ABcde$

ESOP Expression is: $de \# CaB \# bDE \# bDEAC \# bc \# eBCdA \# a \# adceb \# cDBaE$

FPRME is: $edc \# edb \# ecb \# dcb \# eda \# eca \# dca \# eba \# dba \# cba$

DS28. Minterms are: 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 16, 17, 18, 20, 24, -1

SOP Expression is: $abcE + abDe + abCd + aBcd + Abcd + abdE + acDe + bcDe + aCde + bCde + Bcde$

ESOP Expression is: $bde \# AcE \# bda \# eCa \# eBDa \# cB \# AcEBD \# cD$

FPRME is: $1 \# e \# eD \# eC \# DC \# eB \# DB \# CB \# eA \# DA \# CA \# BA \# DCBA \# eDCBA$

DS29. Minterms are: 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 17, 18, 19, 20, 21, 22, 24, 25, 26, 28, -1

SOP Expression is: $abDE + aCdE + aCDe + aBcE + BcDe + BCde + AbcE + AbDe + AbCd + ABcd$

ESOP Expression is: $Ab \# Ac \# eCB \# aEd \# bcE \# Db \# DeA \# Dc$

FPRME is: $ED \# EC \# DC \# EB \# DB \# CB \# EA \# DA \# CA \# BA$

DS30. Minterms are: 7, 11, 13, 14, 15, 19, 21, 22, 23, 25, 26, 27, 28, 29, 30, -1
SOP Expression is: $aCDE + BcDE + BCdE + BCDe + AbDE + bCDE + AbCE + AbCD + ABcE + ABcD + ABCd$
ESOP Expression is: $Ca \# aEBd \# EBC \# Cbe \# Cbeda \# ADb \# ADe \# Cd$
FPRME is: $1 \# ed \# ec \# dc \# eb \# db \# cb \# edcb \# A \# eA \# dA \# cA \# bA \# edcbA$

Positive Function(31-33):

DS31. Minterms are: 4, 6, 7, 12, 14, 15, -1
SOP Expression is: $Bd + BC$
ESOP Expression is: $B \# BDc$
FPRME is: $B \# DcB$

DS32. Minterms are: 4, 7, 12, 15, -1
SOP Expression is: $Bcd + BCD$
ESOP Expression is: $Bc \# BD$
FPRME is: $dB \# CB$

DS33. Minterms are: 4, 6, 7, 13, 15, 20, 22, 23, 29, 31, -1
SOP Expression is: $bCe + BCE + bCD$
ESOP Expression is: $CbED \# CB \# Ce$
FPRME is: $EC \# Cb \# EDCb$

Vacuous Function(DS34-DS36):

DS34. Minterms are: 24, 25, 26, 27, 28, 29, 30, 31, -1
SOP Expression is: AB
ESOP Expression is: AB
FPRME is: BA

DS35. Minterms are: 2, 7, 8, 13, 18, 23, 24, 29, -1
SOP Expression is: $bcDe + bCDE + Bcde + BCdE$
ESOP Expression is: $dC \# de \# bC \# be$
FPRME is: $eD \# DC \# eB \# CB$

DS36. Minterms are: 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 29, 31, -1

SOP Expression is: $aE + bcE + BCE$

ESOP Expression is: $EAC \# E \# EAB$

FPRME is: $E \# ECA \# EBA$

Unate Function(DS37-DS39):

DS37. Minterms are: 6, 12, 13, 14, 15, -1

SOP Expression is: $AB + BCd$

ESOP Expression is: $BaCd \# BA$

FPRME is: $dCB \# BA \# dCBA$

DS38. Minterms are: 12, 13, 14, 15, 22, 28, 29, 30, 31, -1

SOP Expression is: $BC + ACDe$

ESOP Expression is: $BC \# bCeDA$

FPRME is: $CB \# eDCA \# eDCBA$

DS39. Minterms are: 1, 3, 4, 5, 6, 7, 12, 13, 14, 15, 17, 18, 19, 20, 21, 22, 23, 28, 29, 30, 31, -1

SOP Expression is: $C + bE + AbD$

ESOP Expression is: $C \# bEc \# AbDec$

FPRME is: $1 \# c \# Ecb \# DcbA \# EDcbA$

Odd Parity Function(DS40-DS42):

DS40. Minterms are: 1, 2, 4, 7, 8, 11, 13, 14, 16, 19, 21, 22, 25, 26, 28, 31, -1

SOP Expression is: $abcdE + abcDe + abCde + abCDE + aBcde + aBcDE + aBCde + aBCDE + Abcde + AbcDE + AbCde + AbCDE + ABcde + ABcDE + ABCde + ABCDE$

ESOP Expression is: $a \# b \# e \# D \# c$

FPRME is: $E \# D \# C \# B \# A$

DS41. Minterms are: 1, 2, 4, 7, 8, 11, 13, 14, 16, 19, 21, 22, 25, 26, 28, 31, 32, 35, 37, 38, 41, 42, 44, 47, 49, 50, 52, 55, 56, 59, 61, 62, -1

SOP Expression is: $abcdeF + abcdeF + abcDef + abcDEF + abCdef + abCdeF + abCDeF + abCDef + aBcdef + aBcdeF + aBcDeF + aBcDEF + aBCdef + aBCdeF + aBCDeF + aBCDef + aBCDEF + Abcdef + AbcdeF + AbcDeF + AbcDEF + AbCdef + AbCdeF + AbCDeF + AbCDef + AbCDEF +$

$AbCDEF + ABcdeF + ABcdEf + ABcDef + ABcDEF + ABCdef + ABCdEF + ABCDeF + ABCDEF$

ESOP Expression is: $a \# B \# e \# f \# D \# c$

FPRME is: $F \# E \# D \# C \# B \# A$

DS42. Minterms are: 1, 2, 4, 7, 8, 11, 13, 14, 16, 19, 21, 22, 25, 26, 28, 31, 32, 35, 37, 38, 41, 42, 44, 47, 49, 50, 52, 55, 56, 59, 61, 62, 64, 67, 69, 70, 73, 74, 76, 79, 81, 82, 84, 87, 88, 91, 93, 94, 97, 98, 100, 103, 104, 107, 109, 110, 112, 115, 117, 118, 121, 122, 124, 127, -1

SOP Expression is: $abcdefG + abcdeFg + abcdEfg + abcdeFG + abcDefg + abcDeFG + abcDEfg + abcDEFg + abCdefg + abCdeFG + abCdEfg + abCdEFg + abCDefg + abCDeFG + abCdeFg + abCDEfg + abCDEFg + aBcdefg + aBcdeFG + aBcdEfg + aBcdEFg + aBcDefg + aBcDeFG + aBcdeFg + aBcDEfg + aBcDEFg + aBCdefg + aBCdeFG + aBCdEfg + aBCdEFg + aBCDeFg + aBCDeFG + Abcdefg + AbcdeFG + AbcdEfg + AbcdEFg + AbcDefg + AbcDeFG + AbcdeFg + AbcDEfg + AbcDEFg + AbCdefg + AbCdEfg + AbCdEFG + AbCDefg + AbCDeFG + AbCdeFg + AbCDEfg + AbCDEFg + ABcdefg + ABcdeFG + ABcdEfg + ABcdEFg + ABcDefg + ABcDeFG + ABcdeFg + ABcDEfg + ABcDEFg + ABCdefg + ABCdeFG + ABCdEfg + ABCdEFg + ABCDeFg + ABCDeFG + ABCdeFg + ABCDEfg + ABCDEFg$

ESOP Expression is: $a \# B \# F \# E \# g \# d \# c$

FPRME is: $G \# F \# E \# D \# C \# B \# A$

Even Parity Function(DS43-DS45):

DS43. Minterms are: 0, 3, 5, 6, 9, 10, 12, 15, 17, 18, 20, 23, 24, 27, 29, 30, -1

SOP Expression is: $abcde + abcDE + abCdE + abCDe + aBcdE + aBcDe + aBCde + aBCDE + AbcdE + AbcDe + AbCde + AbCDE + ABcde + ABcDE + ABCde + ABCDE$

ESOP Expression is: $a \# b \# E \# D \# c$

FPRME is: $e \# D \# C \# B \# A$

DS44. Minterms are: 0, 3, 5, 6, 9, 10, 12, 15, 17, 18, 20, 23, 24, 27, 29, 30, 33, 34, 36, 39, 40, 43, 45, 46, 48, 51, 53, 54, 57, 58, 60, 63, -1

SOP Expression is: $abcdef + abcdeF + abcDeF + abcDEf + abCdeF + abCdEf + abCDef + abcDEF + aBcdeF + aBcdEf + aBcDef + aBcDEF + aBCdef + aBCdEF + aBCDeF + aBCDEF + AbcdeF + AbcdEf + AbcDef + AbcDEF + AbCdef + AbCdEF + AbCDeF + AbCDEF + ABcdef + ABcdEf + ABcDef + ABcDEF + ABCdef + ABCdEF + ABCDeF + ABCDEF$

+ ABCDEF

ESOP Expression is: a # B # e # F # D # c

FPRME is: f # E # D # C # B # A

DS45. Minterms are: 0, 3, 5, 6, 9, 10, 12, 15, 17, 18, 20, 23, 24, 27, 29, 30, 33, 34, 36, 39, 40, 43, 45, 46, 48, 51, 53, 54, 57, 58, 60, 63, 65, 66, 68, 71, 72, 75, 77, 78, 80, 83, 85, 86, 89, 90, 92, 95, 96, 99, 101, 102, 105, 106, 108, 111, 113, 114, 116, 119, 120, 123, 125, 126, -1

SOP Expression is: abcdefg + abcdeFG + abcdEfG + abcdEFg + abcDefG + abcDeFg + abcDEfg + abcDEFG + abCdefG + abCdeFg + abCdEfg + abCdEFG + abCDefg + abCDEfg + abCDEFg + abCDEFG + aBcdefG + aBcdeFG + aBcdEfg + aBcdEFG + aBCdefg + aBCdeFG + aBCdEfg + aBCdEFG + aBCDefg + aBCDeFG + aBCdEfg + aBCdEFG + AbcdefG + AbcdeFg + AbcdEfg + AbcdEFG + AbcDefg + AbcDeFG + AbcDEfg + AbcDEFg + AbCdefg + AbCdeFG + AbCdEfg + AbCdEFG + AbCDefg + AbCDeFG + AbCDEfg + AbCDEFG + ABcdefG + ABcdeFG + ABcdEfg + ABcdEFG + ABcDefg + ABcDeFG + ABcDEfg + ABcDEFg + ABcDEFG + ABCdefg + ABCdeFG + ABCdEfg + ABCdEFG + ABCDefg + ABCDeFG + ABCDEfg + ABCDEFG + ABCDeFG + ABCDEfg + ABCDEFg

ESOP Expression is: a # B # F # E # G # d # c

FPRME is: g # F # E # D # C # B # A

Majority Function(DS46):

DS46. Minterms are: 7, 11, 13, 14, 15, 19, 21, 22, 23, 25, 26, 27, 28, 29, 30, 31, -1

SOP Expression is: CDE + BDE + BCE + BCD + ADE + ACE + ACD + ABE + ABD + ABC

ESOP Expression is: ECd # BdAE # BdAC # ECab # D # Dab # cDeA # cDeB

FPRME is: EDC # EDB # ECB # DCB # EDCB # EDA # ECA # DCA # EDCA # EBA # DBA # EDDBA # CBA # ECBA # DCBA

DS47. Minterms are: 0, 1, 3, 5, 7, 10, 13, 15, 16, 19, 20, 22, 24, 27, 29, 31, 32, 35, 36, 38, 39, 40, 41, 43, 45, 46, 47, 49, 50, 51, 54, 56, 57, 58, 59, 61, 64, 68, 69, 70, 74, 76, 78, 79, 81, 84, 90, 92, 94, 95, 97, 99, 104, 106, 107, 109, 113, 116, 118, 119, 124, 125, 127, 129, 131, 136, 138, 140, 143, 149, 150, 153, 159, 162, 163, 164, 169, 173, 176, 177, 179, 181, 184, 185, 186, 189, 195, 196, 199, 200, 201, 205, 206, 207, 209, 217, 219, 220, 225, 226, 227, 229, 230, 234, 235, 236, 239, 240, 241, 243, 245, 247, 248, 249, 251, 252,

254, 255, 257, 258, 259, 261, 264, 267, 270, -1

SOP Expression is: acDFhl + aBDEgh + acdefgl + bcdefhl + acdFGHI + abcDeGH + abDFGhl + acDEFgi + abCdeGi + abCdFHi + aCdeFGH + aCEfghl + aCEFGhi + aBEFghl + aBDefgH + aBCEfgl + aBCDFhl + acdEfGHi + acDefGhi + aCdefGhi + aBcdeFhi + aBcEfGhl + aBCDefHi + aBCDFGhi + AbcdefgH + AbcdegHI + AbcdeFghi + AbcdeFGHi + abcEgHI + abDegHI + aCDEGHI + abcdgghi + acdeFgHi + abDEfGhi + abcDeghi + abcDEfHi + abCDeFgi + abCDEfGi + aBCdefHI + aBCdeFgh + abcdefl + abcdeGI + abcdFGI + abDFGHI + acDfgHI + abcDEgl + aCDefgl + aCdegHI + aCDeFgH + aBDfgHI + aBCDFHI + abdefghi + abdefGhl + abcdEghi + aBCdeFhl + aBCDEFGi

ESOP Expression is: aBehIDf # abfhigE # acdHeF # aigchdB # aDh # acGihD # ahbFGI # acEhdgFI # acGihDeF # aBIC # ahEgBCI # dlhBa # aDhei # acDeFglhb # aBGeIhDFC # adGIF # aEbdhf # aCGhF # aefiCd # aBdeFh # aefiCBgd # cedIhB # aCDFhb # GcaIBdHf # bceg # bcedIhAGF # aHFcB # aHbcEi # acEIhBFG # agHDE # bchgedf # aHeB # agICHDF # agHDI # aCHibd # bcegAD # aBDGEC # cedIhA # gcadh # aGfceH # aCh # aBdfH # aBDFHGe # aCHfiE # aHeBig # adfHGie # aCDEGHI # aefiCBgH # AbcdeiFH # IbcdegHa

FPRME is: hecb # ihecb # gecb # hgecb # iFecb # hFecb # ihgFecb # heDcb # iheDcb # geDcb # hgeDcb # iFeDcb # hFeDcb # ihgFeDcb # hga # ihga # ihFa # gFa # igFa # hgFa # ea # iea # hea # hgea # iFea # hFea # igFea # Da # iDa # hgDa # iFDa # hFDa # gFDa # ihgFDa # ieDa # geDa # igeDa # FeDa # ihFeDa # gFeDa # ica # hca # igca # ihgca # Fca # igFca # eca # geca # ihFeca # gFeca # Dca # ihDca # gDca # hgDca # FDca # iFDca # hFDca # ihFDca # gFDca # ihgFDca # hgeDca # ihgeDca # FeDca # iFeDca # igFeDca # ihgFeDca # ihba # ihgba # Fba # hFba # ihFba # hgFba # eba # igeba # hgeba # iDba # hDba # gDba # FDba # hFDba # eDba # iheDba # geDba # igeDba # FeDba # hFeDba # gFeDba # igFeDba # ihgFeDba # hcba # ihcba # gcba # igcba # ihgcba # Fcba # hFcba # icba # ihecba # gecba # igecba # ihgecba # Fecba # ihFecba # gFecba # igFecba # ihgFecba # hDcba # igDcba # hgDcba # FDcba # iFDcba # ihFDcba # gFDcba # igFDcba # hgFDcba # ihgFDcba # eDcba # geDcba # hgeDcba # ihgeDcba # iFeDcba # ihFeDcba # igFeDcba # hgFeDcba # ihgFeDcba

DS48. Minterms are: 1, 2, 5, 6, 7, 10, 12, 14, -1

ESOP Expression is: aD # BcdA # aDbC # dC

DS49. Minterms are: 2, 5, 7, 10, 13, 16, 18, 24, -1

ESOP Expression is: aCE # aCEDB # ce # ceda # ceADB

DS50. Minterms are: 2, 5, 13, 16, 24, 32, 36, 45, -1

ESOP Expression is: $bcdfaE \# bceA \# aBdef \# AbcedF \# bDeF$

DS51. Minterms are: 5, 10, 17, 23, 33, 47, 67, 78, -1

ESOP Expression is: $adfGce \# bcDFga \# adfGb \# aBcDEFG \# AbcdeFG \# bcDFgE \# adGbCE$

DS52. Minterms are: 10, 18, 34, 69, 90, 110, 129, 170, -1

ESOP Expression is: $abfGhd \# abfGhec \# aBcdeFgH \# AbcdefgH \# aBcDEfGh \# aBCdEFGh \# bfGhdEC$

DS53. Minterms are: 10, 18, 36, 70, 100, 129, 257, 450, -1

ESOP Expression is: $abcdgHie \# abDefGhi \# abCdefGHi \# cdefghIa \# cdefghIb \# ABCdefgHi \# abcdgHif$

DS54. Minterms are: 0, 1, 2, 3, 4, 5, 6, 8, 9, 11, 12, 13, 15, 17, 19, 22, 23, 25, 26, 28, 29, 30, -1

ESOP Expression is: $ECb \# BDea \# AE \# ECbda \# 1 \# AedBC \# AcDb \# Ad$

DS55. Minterms are: 5, 8, 19, 1, 3, 28, 30, 26, 23, 17, 12, 2, 4, 9, 6, 0, 25, 11, 29, 22, 15, 13, -1

ESOP Expression is: $AB \# DBea \# bDCa \# 1 \# bDCe \# cBAd \# Ae \# ACeD$

DS56. Minterms are: 0, 2, 4, 6, 8, 12, 17, 19, 23, 29, 25, 28, 1, 5, 13, 9, 22, 30, 26, 3, 15, 11, -1

ESOP Expression is: $DBae \# AB \# bCDE \# 1 \# bCDA \# cBAd \# Ae \# AECD$

DS57. Minterms are: 0, 4, 12, 8, 28, 22, 30, 26, 2, 6, 17, 29, 25, 1, 5, 13, 9, 19, 23, 3, 15, 11, -1

ESOP Expression is: $Aedc \# DB \# bDEA \# DA \# DE \# bCA \# bDEc \# 1$

DS58. Minterms are: 2, 13, 8, 23, 29, 3, 5, 12, 22, 25, 1, 4, 30, 0, 26, 6, 15, 9, 19, 28, 11, 17, -1

ESOP Expression is: $AB \# BDea \# bCDA \# 1 \# AECD \# bCDe \# BAcd \# Ae$

DS59. Minterms are: 0, 2, 6, 9, 10, 15, 17, 19, 21, 22, 27, 29, 31, 35, 37, 39, 41, 43, 48,

51, 53, 57, 59, 61, 68, 69, 71, 79, 81, 86, 83, 89, 91, 95, 97, 101, 106, 107, 109, 112, 118, 162, 164, 167, 169, 184, 186, 189, 197, 199, 201, 214, 215, 219, 221, 224, 228, 231, 237, 239, 241, 245, 246, 247, 249, 251, 253, 255, 257, 259, 261, 263, 265, 267, 268, 271, 273, 275, 277, 279, 281, 283, 287, 291, 293, 297, 299, 301, 312, 314, 317, 321, 325, 331, 332, 337, 339, 341, 343, 347, 349, 354, 359, 367, 368, 372, 380, 384, 385, 389, 394, 395, 397, 399, 401, 409, 421, 425, 429, 460, 468, 471, 501, 507, 513, 517, 518, 520, 522, 529, 534, 550, 556, 557, 559, 568, 569, 571, 573, 580, 581, 587, 589, 592, 593, 594, 595, 596, 600, 604, 609, 614, 617, 620, 628, 631, 651, 657, 660, 675, 678, 681, 682, 685, 693, 694, 697, 699, 712, 713, 714, 717, 721, 725, 734, 735, 737, 740, 741, 747, 749, 781, 782, 783, 981, 964, 965, 968, 975, 976, 981, 982, 985, 987, 988, 990, 994, 995, 997, 999, 1012, 1013, 1017, 1021, 1024, 1028, 1069, 1127, 1158, 1159, 1169, 1200, 1249, 1250, 1254, 1259, 1300, 1347, 1349, 2000, 2014, 2145, 2489, 2510, 2678, 2900, 3001, 5012, 5013, 5094, 5095, 5099, 5100, 5189, 6000, 6048, 6049, 6148, 6149, 6258, 6259, 6262, 7001, 7002, 7008, 7009, 7501, 7509, 7519, 8000, 8004, 8018, 9000, 9999, -1,

SOP Expression is: abcdeFgikmN + abcdfghikLMn + abcdeghjKlmN + abcdfgHijkLm + abcdefHijkLN + abcdefgHiJIN + abcdefHjKLmN + abcdefGHIJmN + abcdeFgiKIMN + abcdeFhiJlmN + abcdeFijKlmN + abcdeFhIJKIN + abcdeFghIjkl + abcdeFghijmn + abcdefgIJklmn + abcdefGHijklmN + abcdefGHijKIM + abcdefGIjklMn + abcdefGhIJKln + abcdfGHijKlmN + abcdefGHJKLMn + abcdefGHIjkmn + abcdeGHIJKIMN + abcdeFgijKLmn + abcdeFghIJKln + abceFgHijkLmN + abcdefHijKLmn + abcdeFgHiJLmN + abcdeFgHijLMN + abcdeFgHIJLmn + abcdeFGhijklm + abcdeFGhijKIM + abcdeFghijKln + abcdeFgIjklMn + abcdeFgIjKLmn + abcdeFghIjKLN + abcdeFghIJKlm + abcdeFgHjklMn + abcdeFgHijlmN + abcdeFghIklLMn + abcdeFgHijKln + abcdeGHijKlmn + abcdeFgHijKLM + abcdeFghijKLN + abcdeFghijKLM + abcdeFGHijKLm + abcEFGHiJklmn + abcdeFGHiJLMn + abcdeFGHIJKIN + abcdeFGHIJKLn + abcEFGHiJKLMn + abcdeFGHIjkIM + abcdeFGHIJKLm + abcDefghijkmn + abcDefGhijklM + abcDefGHIjkmn + abCdFgHjklmN + aBcdEFGhiJkLm + aBcdEFGHIjkLM + aBcDEFghijklm + aBCdefghijKlM + aBCdefgHIJKIM + aBCdefgHIJkMn + aBCdeFGHijklm + aBCDEFghijkmn + abcdefGhijklMn + abcdeFgHijklMn + abcdeFGHiJkLmn + abcdeFghijKIMN + abcdeFgHijKLMN + abcdeFghijKIMN + abcdeFghIjKLmn + abcdeFghIjklMn + abcdeFghIjKIMN + abcdeFghIjKLmN + abcdeFgHijKIMN + abcdeFGHijKLMN + abcDefghIjKLmN + abcDefghIjklMn + abcDefGhiJklmN + abcDefGhiJklmn + abcDefGHIjklmN + abcDefGHIjKIMN + abcDefGhiJkLmn + abcDefGhijKIMN + abCdeFGHijKLMn + abCdEfgHijKLMn + abCdeFGHiJkLmn + aBcdEFGHIjKIMN + aBcdEFGHIJKLmn + aBcDefghijKlMn + aBcDEFghijklmN +

aBCdEFgHIJKlmN + aBCdEFgHiJKlMn + aBCDeFgHijKLmN + aBCDeFgHiJkLmN +
aBCDeFgHiJKLMN + aBCDEFgHiJklMn + AbcdEFghIjKlmn + AbcDEFghijKLMN +
abcdefgiKLMN + abcdefghJIMN + abcdefghJLmN + abcdeghIjIMN + abcdefGHILMN
+ abcdeFhIjLmN + abcdefghijkln + abcdefghIjklN + abcdefgHIjLmN + abcdFghIjKlmN
+ abcdeGHIjKLMN + abcdFghIjKLmN + abcdeFGhijKLN + abcdeFghijkmN +
abcdEfHijKLmN + abcdEGHiJkLmN + abcdEfGHIjkmN + abcdEFGHIjklN +
abcdEFGHIJKmN + abcdegiJklN + abcdeFghikN + abcdeFghiMN + abcdeFgiJkN +
abcdefghIKIN + abcdehljKlmN + abcdfghIJKmN + abcdeFhijkmN + abcdEFGHkLmN
+ abcdefghijMn + abcdefhljKLMN + abcdefhljKLMN + abcdeghIJKLmN +
abcdefHiJKIMN + abcdegHIJklmn + abcdFghIjklmn + abcdeGHIjKLMN +
abcdEfhljKLMN

BIBLIOGRAPHY

- [1] A. Tran and E. Lee, Generalisation of Tristate Map and A Composition Method for Minimization of Reed-Muller Polynomials in Mixed Polarity., IEE Proceedings-E. Vol. 140, No. 1, Jan. 1993, pp. 59-64.
- [2] Arpad Barna, and Dan I. Porat, Integrated Circuits in Digital Electronics, John Wiley & Sons Inc., 1973.
- [3] Development of an Improved Logic Minimization Algorithm with Software Implementation of Quine-McCluskey and Algorithm-S Method, A dissertation submitted by roll No. 4302, Department of Applied Physics & Electronics, DU, Dhaka, August, 1993.
- [4] Dietmeyer, D.L., Logic Design of Digital Systems., Massachusetts, Allyn and Bacon, Inc., 1979.
- [5] Edited by Tsutomu Sasao, Logic Synthesis and Optimization, Kyushu Institute of Technology, Iizuka 820, Japan.
- [6] Edward J. McCluskey, Logic Design Principles., New Jersey, Prentice Hall, 1986, pp. 51, Chap 2.
- [7] Fredrick J. Hill, Gerald R. Peterson, Digital Systems Hardware Organisation and Design, John Wiley & Sons Inc., 1987.

- [8] Hasan, H.M. and Mottalib, M.A., Design of Minimized Logic Networks Using EXOR and AND Gates by a Computer with a Small Memory Space., Accepted for publication in the Bangladesh Journal of Scientific and Industrial Research, BCSIR, Dhaka.
- [9] M. Morris Mano, Digital Design, Prentice/Hall International, Inc., 1984.
- [10] M. Perkowski and M. Chrzanowska-Jespe, An Exact Algorithm to Minimize Mixed-Radix Exclusive Sums of Products for Incompletely Specified Boolean Functions, Proc. ISCAS, pp. 1652-1655, June 1990.
- [11] Marinkovic, S.B. and Tosic, Z., Algorithm for Minimal Polarized Polynomial Form Determination., IEEE Trans. Comput. Vol. C-23, pp. 1313-1315, Dec. 1974.
- [12] Minato. S., Ishiura. N and Yajima. S., Paper 3.3, 27th ACM/IEEE Design Automation Conference.
- [13] Mottalib, A., Design of Easily Testable PLAs and Their Testing Algorithm., Thesis for the degree of Doctor of Philosophy (I.I.T Kharagpur),, March 1992.
- [14] Mottalib, M.A. and Md. Hasinur Rahman, A logic Minimization Technique Using Reed-Muller Cannonic Expansion with Software Implementation, Journal of the Bangladesh Electronics Society, pp. 9-13, Vol. 3, No. 1, 1993.
- [15] Mukhopadhyay, A. and Schmitz, G., Minimization of Exclusive OR and Logical Equivalence of Switching Circuits, IEEE Trans., 1970, C-19, pp. 132-140.
- [16] N. Koda and T. Sasao, A Minimization Method for AND-EXOR Expressions Using Lower Bound Theorem, (in Japanese), Trans. IEICE(to be published).

- [17] Ph.W. Besslich, Dr.-Ing., Efficient Computer Method for EXOR Logic Design, IEE Proceedings, Vol.130, Pt.E, No.6, Nov. 1983.
- [18] S. Swamy, On Generalized Reed-Muller Expansions, IEEE Trans. Comput., Vol. C-21, pp. 1008-1009, Sept. 1972.
- [19] S.M. Reedy, Easily Testable Realizations for Logic Functions, IEEE Trans. Comput., Vol. C-21, pp. 1183-1188, Nov., 1972.
- [20] Saluja K. and Ong E.H., Minimization of Reed-Muller Canonic Expansion, IEEE Trans. Comput., Vol. C-28, No. 7, July 1979.
- [21] Shannon, C.E., A symbolic Analysis of Relay and Switching Circuits, Trans. Am. Inst. Electr. Eng., Vol. 57, pp. 713-723, 1938.
- [22] T. Sasao, On the Complexity of Some Classes of AND-EXOR Expressions., IEICE Technical Report FTS 91-39, Oct. 1991.
- [23] T. Sasao, EXMIN2: A Simplification Algorithm for Exclusive-or-sum of Products Expressions for Multiple-Valued Input Two-Valued Output Functions, IEEE Trans on CAD (to be published).
- [24] T. Sasao, Optimization of Pseudo-Kronecker Expressions Using Multiple-Place Decision Diagrams, IEICE Trans. INF. & SYST., Vol. E76-D, No.5, May 1993.
- [25] T. Sasao and P. Besslich, On the Complexity of MOD-2 Sum PLA's., IEEE Trans. on Comput., Vol. 39, No. 2, pp. 262-266, Feb. 1990.

- [26] Tran, A., Tri-state Map for the Minimization of Exclusive-OR Switching Functions, IEE Proc. E, 1989, 136, (1), pp.16-21.
- [27] Tran, A., Graphical Method for the Conversion of Minterms to Reed-Muller Coefficients and the Minimization of Exclusive-OR Switching Functions, IEE Proc. E, 1987, 134, (2), pp. 93-99.
- [28] Tran, A., and Wang, J., Decomposition Method for Minimization of Reed-Muller Polynomials in Mixed Polarity, IEE Proc. E, 1993, 140, (1), pp. 65-68.
- [29] Tsutomu Sasao, Optimization of Multiple-Valued AND-EXOR Expressions Using Multiple-Place Decision Diagrams, Kyushu Institute of Technology, Iizuka 820, Japan.
- [30] Whitesitt, J.E., Boolean Algebra and Its Applications, Addison-Wesley Publishing Company, Inc., Reading, Mass., 1961.
- [31] Wood, R.E., Switching Theory, New York, McGrawhill, 1968, pp. 72-73, Chap.3.
- [32] Wu X., Chen X., and Hurst S.L., Mapping of Reed-Muller Coefficients and the Minimization of Exclusive-OR Switching Functions., IEE Proc. E, Comput. & Digital Tech., 1982, 129, (1), pp. 15-20.

