# LOGIC SYNTHESIS AND OPTIMISATION USING REED-MULLER EXPANSIONS 

Lynn Mhairi MCKenzie

A thesis submitted in partial fulfilment of the requirements of Napier University for the degree of Doctor of Philosophy in the Department of Electrical, Electronic and Computer Engineering.

## Abstract

This thesis presents techniques and algorithms which may be employed to represent, generate and optimise particular categories of Exclusive-OR Sum-Of-Products (ESOP) forms. The work documented herein concentrates on two types of Reed-Muller (RM) expressions, namely, Fixed Polarity Reed-Muller (FPRM) expansions and KROnecker (KRO) expansions (a category of mixed polarity RM expansions). Initially, the theory of switching functions is comprehensively reviewed. This includes descriptions of various types of RM expansion and ESOP forms. The structure of Binary Decision Diagrams (BDDs) and Reed-Muller Universal Logic Module (RM-ULM) networks are also examined.

Heuristic algorithms for deriving optimal (sub-optimal) FPRM expansions of Boolean functions are described. These algorithms are improved forms of an existing tabular technique [1]. Results are presented which illustrate the performance of these new minimisation methods when evaluated against selected existing techniques. An algorithm which may be employed to generate FPRM expansions from incompletely specified Boolean functions is also described. This technique introduces a means of determining the optimum allocation of the Boolean 'don't care' terms so as to derive equivalent minimal FPRM expansions.

The tabular technique [1] is extended to allow the representation of KRO expansions. This new method may be employed to generate KRO expansions from either an initial incompletely specified Boolean function or a KRO expansion of different polarity. Additionally, it may be necessary to derive KRO expressions from Boolean Sum-Of-Products (SOP) forms where the product terms are not minterms. A technique is described which forms KRO expansions from disjoint SOP forms without first expanding the SOP expressions to minterm forms.

Reed-Muller Binary Decision Diagrams (RMBDDs) are introduced as a graphical means of representing FPRM expansions. RMBDDs are analogous to the BDDs used to represent Boolean functions. Rules are detailed which allow the efficient representation of the initial FPRM expansions and an
algorithm is presented which may be employed to determine an optimum (sub-optimum) variable ordering for the RMBDDs. The implementation of RMBDDs as RM-ULM networks is also examined.

This thesis is concluded with a review of the algorithms and techniques developed during this research project. The value of these methods are discussed and suggestions are made as to how improved results could have been obtained. Additionally, areas for future work are proposed.

## List of Contents

List of Illustrations ..... vii
List of Accompanying Material ..... ix
Acknowledgements ..... x
Declaration ..... xi
List of Abbreviations ..... xii
Chapter 1 Introduction ..... 1
Chapter 2 Theory and Definitions ..... 7
2.1 Logic Functions ..... 7
2.2 Algebra of GF(2) ..... 8
2.3 Fixed Polarity Reed-Muller Expansions ..... 9
2.4 Shannon Expansion Theorem and Exclusive-OR Sum-of- Products Forms ..... 17
2.5 Classes of Exclusive-OR Sum-of-Products Expansions ..... 21
2.5.1 Kronecker Expansions ..... 21
2.5.2 Pseudo Reed-Muller Expansions and Pseudo Kronecker Expansions ..... 25
2.5.3 Generalised Reed-Muller Expansions ..... 31
2.5.4 Exclusive-OR Sum-of-Products Expansions ..... 31
2.6 Binary Decision Diagrams ..... 32
2.7 Reed-Muller Universal Logic Module Networks ..... 37
2.8 Summary ..... 44
Chapter 3 Logic Minimisation Using Fixed Polarity Reed-Muller Expansions ..... 45
3.1 Review of Techniques for Fixed Polarity Reed-Muller Expansions ..... 46
3.2 Minimisation Techniques for Fixed Polarity Reed-Muller Expansions ..... 55
3.2.1 Review of Tabular Techniques ..... 55
3.2.2 Full Gains Minimisation Algorithm ..... 62
3.2.3 Pre-treatment Techniques for the Full Gains Minimisation Algorithm ..... 70
3.3 Summary ..... 74
Chapter 4 Fixed Polarity Reed-Muller Expansions of Incompletely Specified Boolean Functions ..... 78
4.1 Fixed Polarity Reed-Muller Expansions of Incompletely Specified Boolean Functions ..... 78
4.2 Review of Techniques for Deriving Fixed Polarity Reed- Muller Expansions of Incompletely Specified Boolean Functions ..... 80
4.3 Minimisation of Fixed Polarity Reed-Muller Expansions using Unspecified Product Terms ..... 84
4.4 Tabular Technique for Deriving Fixed Polarity Reed-Muller Expansions from Incompletely Specified Boolean Functions ..... 91
4.5 Determining Minimal FPRM Expansions of Incompletely Specified Boolean Functions ..... 98
4.6 Summary ..... 103
Chapter 5 Generating Kronecker Expansions Using a Tabular Technique ..... 107
5.1 Review of Optimisation Techniques for Kronecker Expansions and Exclusive-OR Sum-of-Products Forms ..... 107
5.2 Tabular Technique for Generating Kronecker Expansions ..... 115
5.3 Tabular Technique for Generating Kronecker Expansions from Incompletely Specified Boolean Functions ..... 132
5.4 Summary ..... 133
Chapter 6 Generating Kronecker Expansions from Reduced Boolean Sum-of-Products Forms ..... 134
6.1 Review of Techniques for Deriving Exclusive-OR Sum-of- Products Forms ..... 134
6.2 Tabular Techniques for Generating Kronecker Expansions from Reduced Boolean Sum-of-Products Forms ..... 136
6.2.1 Reduced Boolean Sum-of-Products Forms ..... 136
6.2.2 Extended Tabular Technique to Derive Positive Polarity Reed-Muller Expansions from Disjoint Reduced Boolean Sum-of-Products Forms ..... 139
6.2.3 Extended Tabular Technique to Derive Kronecker Expansions from Disjoint Reduced Boolean Sum-of- Products Forms ..... 147
6.3 Summary ..... 153
Chapter 7 Graphical Representation of Fixed Polarity Reed-Muller Expansions ..... 154
7.1 Multi-level Logic Synthesis Techniques ..... 155
7.2 Reed-Muller Binary Decision Trees and Diagrams ..... 169
7.3 Minimising RORMBDD Representations through Variable Ordering ..... 177
7.3.1 Variable Ordering Algorithm for deriving RORMBDDs ..... 179
7.4 Physical Implementation of RORMBDDs using Reed-Muller Universal Logic Modules ..... 187
7.5 The Reed-Muller Factored Form and Gate Level Implementation of RORMBDDs ..... 190
7.6 Summary ..... 192
Chapter 8 Conclusions ..... 193
8.1 Review of Algorithms and Techniques ..... 193
8.2 Further Research ..... 195
List of References ..... 197

## List of Illustrations

Figure 2.1: Basic connectives of $\mathrm{GF}(2)$ algebra and the equivalent logical operators. ..... 9
Figure 2.2: Classes of Exclusive-OR sum-of-products expansions and their relationships. ..... 22
Figure 2.3: Ordered Binary Decision Tree of $n$ variable Boolean function ..... 33
Figure 2.4: Non-terminal node representing function variable $x_{j}$ ..... 33
Figure 2.5: (a) OBDT of the Boolean function, (b) OBDD, showing redundant nodes and isomorphic subfunctions, (c) ROBDD of the Boolean function. ..... 36
Figure 2.6: Symbol of a RM-ULM(c) ..... 38
Figure 2.7: Circuits and symbols of (a) RM-ULM(1) and (b) RM- ULM(2) ..... 39
Figure 2.8: RM-ULM(2) implementation of the FPRM expansion given in Equation (2.27) ..... 41
Figure 2.9: RM-ULM network ..... 42
Figure 2.10: Tree network implementing the FPRM expansion given in Equation (2.30). ..... 43
Figure 2.11: Cascade network implementing the FPRM expansion given in Equation (2.31). ..... 44
Figure 3.1: Percentage of randomly generated Boolean functions for which the minimisation algorithms formed optimum FPRM expansions. (Boolean functions of 4 variables, 3-13 minterms ( 1000 Boolean functions per set).) ..... 66
Figure 3.2: Percentage of Boolean functions for which the minimisation algorithms formed optimum FPRM expansions. (All Boolean functions of 4 variables, 3-13 minterms.) ..... 67
Figure 3.3: Percentage of randomly generated Boolean functions for which the minimisation algorithms formed optimum FPRM expansions. (Boolean functions of $4-10$ variables.) ..... 68
Figure 3.4: Percentage of randomly generated Boolean functions for which the minimisation techniques (pre-treated) formed optimum FPRM expansions. (Boolean functions of 4 variables, 3- 13 minterms ( 1000 Boolean functions per set).) ..... 75
Figure 3.5: Percentage of randomly generated Boolean functions for which the minimisation techniques (pre-treated) formed optimum FPRM expansions. (Boolean functions of 4-10 variables.) ..... 76
Figure 4.1: Minimisation of a Boolean function and an ESOP expression using 'don't care' terms. ..... 81
Figure 4.2: Tree-type structure used to determine the optimum use of 'don't care' terms. ..... 86
Figure 4.3: Determining the optimum use of the 'don't care' terms for Example 4.2. ..... 96
Figure 4.4: Percentage of randomly generated incompletely specified Boolean functions for which minimisation algorithms formed optimum FPRM expansions. (Boolean functions of 4-6 variables ( 1000 Boolean functions per set).) ..... 104
Figure 4.5: Time taken for algorithms to optimise sets of 1000 incompletely specified Boolean functions ..... 105
Figure 7.1: Non-terminal node representing function variable $x_{j}$ ..... 170
Figure 7.2: Ordered Reed-Muller Binary Decision Tree of $n$ variable FPRM expansion. ..... 172
Figure 7.3: ORMBDT of a 4-variable RM expansion (polarity 0 ), Example 7.1 ..... 173
Figure 7.4: Merger rule ..... 175
Figure 7.5: Deletion rule ..... 176
Figure 7.6: RORMBDD of Example 7.2 ..... 176
Figure 7.7: RORMBDD of the ESOP expansion. ..... 178
Figure 7.8: RORMBDD of Example 7.3 ..... 187
Figure 7.9: (a) Circuit implementation of a RM-ULM(1), (b) Symbol of a RM-ULM(1), (c) Single node of a RMBDD ..... 188
Figure 7.10: RM-ULM(1) implementation of the RORMBDD of Example 7.3 ..... 189
Figure 7.11: Reduced RM-ULM(1) implementation of the RORMBDD of Example 7.3 ..... 189

## List of Accompanying Material

Copies of the following publications may be found in a pocket attached to the inside back cover of this thesis.

McKenzie, L., Almaini, A.E.A, Miller, J.F., Thomson, P., 'Optimisation of Reed-Muller logic functions', Int. J. Electronics, 75, (3), 1993, pp. 451-466

Xu, L., Almaini, A.E.A., Miller, J.F., McKenzie, L., 'Reed-Muller Universal Logic Module Networks', I.E.E. Proc. E, 140, (2), 1993, pp. 105-108

McKenzie, L., Xu, L., Almaini, A.,
'Graphical Representation of Generalised Reed-Muller Expansions', Proc. IFIP WG 10.5 Workshop On Applications of the Reed-Muller Expansion in Circuit Design, 1993, pp. 181-187

Xu, L., McKenzie, L.
'Multi-level Optimisation of Fixed Polarity Reed-Muller Expansions using Reed-Muller Binary Decision Diagrams',
Proc. I.E.E. Colloquium on Synthesis and Optimisation of Logic Systems, 1994, pp. 3/1-3/4

## Acknowledgements

This project was initially funded by the Science and Engineering Research Council under grant reference GR/F 74288. Additional funding was provided by Napier University.

I would like to express my gratitude for the support offered to me by Professor A. E. A. Almaini, Department of Electrical, Electronic and Computer Engineering, Napier University, under whose patient supervision this project was undertaken and completed. I would also like to thank Professor N. D. Deans, Robert Gordons' University who kindly acted as second supervisor.

I offer my sincere thanks to my friends and former colleagues, Ms Leeka Xu, Mr. Ivan Reading, Dr. Gerald Starrs, Mr. Z. Guan and Mr. John MCMahon, all of Dept. E.E.C.E, Napier University. Our good-natured discussions and their advice proved to be invaluable. Additionally, I would like to acknowledge the generous support given to me by members of the lecturing, technical and secretarial staff of Dept. E.E.C.E., Napier University. Particular thanks go to Dr. Julian Miller, Mr. Peter Thomson, Mr. Michael Williams, Mr. Gavin Wilson and Mr. John Mumford. I would also like to express my gratitude to Dr. D. Cummings, Mrs. T. Thomson and other members of the library staff at Craiglockhart Library, Napier University.

Thanks also to Dr. Jonathan Saul, formerly of Bristol University, for his valued technical help.

I would like to thank Mr. Robert Adamson, Mrs. Janette MCKenzie, Mr. Ian MCKenzie, Ms. Fiona Reid and Dr. Guang Geng who encouraged me throughout this project and also diligently read and corrected this thesis.

Finally, this thesis is dedicated to my parents Ian and Janette McKenzie. Without their unending support and guidance this project could not have been completed nor this thesis written.

## Declaration

I declare that no material contained in this thesis has been used in any other submission for an academic award.

Lynn M. MCKenzie

## List of Abbreviations

| ALT | - Adaptive Logic Tree |
| :--- | :--- |
| BDD | - Binary Decision Diagram |
| BDT | - Binary Decision Tree |
| CLB | - Configurable Logic Block |
| ESOP | - Exclusive-OR Sum-Of-Products |
| EXOR | - EXclusive-OR |
| FDD | - Functional Decision Diagram |
| FPGA | - Field Programmable Gate Array |
| FPRM | - Fixed Polarity Reed-Muller |
| GF(q) | - Galois Field (q) |
| GF(2) | - Galois Field (2) |
| GRM | - Generalised Reed-Muller |
| KFDD | - Kronecker Functional Decision Diagram |
| KRM | - Kronecker Reed-Muller |
| KRO | - KROnecker |
| MCNC | - Microelectronics Center of North Carolina |
| OBDD | - Ordered Binary Decision Diagram |
| OBDT | - Ordered Binary Decision Tree |
| OKFDD | - Ordered Kronecker Functional Decision Diagram |
| OR | - Inclusive-OR |
| ORMBDD | - Ordered Reed-Muller Binary Decision Diagram |
| ORMBDT | - Ordered Reed-Muller Binary Decision Tree |
| PKRM | - Pseudo Kronecker Reed-Muller |
| PLA | - Programmable Logic Array |
| PPRM | - Positive Polarity Reed-Muller |
| PSDKRO | - PSeuDo KROnecker |
| PSDRM | - PSeuDo Reed-Muller |
| RM | - Reed-Muller |
| RMBDD | - Reed-Muller Binary Decision Diagram |
| RMBDT | - Reed-Muller Binary Decision Tree |
| RM-ULM | - Reed-Muller Universal Logic Module |
| RM-ULM(c) | - Reed-Muller Universal Logic Module (c control inputs) |
| RM-ULM(1). - Reed-Muller Universal Logic Module (single control input) |  |
| RM-ULM(2) | - Reed-Muller Universal Logic Module (two control inputs) |
|  | PR |


| ROBDD | - Reduced Ordered Binary Decision Diagram |
| :--- | :--- |
| ROKFDD | - Reduced Ordered Kronecker Functional Decision Diagram |
| RORMBDD | - Reduced Ordered Reed-Muller Binary Decision Diagram |
| SOP | - Sum-Of-Products |
| TDD | - Ternary Decision Diagram |
| XPLA | - EXclusive-OR Programmable Logic Array |

## Chapter 1

## Introduction

The increasing complexity of electronic systems demands high performance integrated circuits which can efficiently and reliably implement their required functions. This, in turn, necessitates the use of sophisticated synthesis tools which aid circuit designers to meet predefined goals, such as area utilisation, performance and testability. Logic synthesis may be considered as comprising of two distinct though not disjoint steps. The first step is to optimise the logic functions by transforming and minimising the representations, independent of the technology being used to realise the functions. The second step is to determine an optimum implementation of the logic functions with the objective of fully exploiting the advantages of the target technology. Generally, synthesis tools will reiterate these two optimisation steps in order to derive efficient implementations.

The algorithms and techniques presented in this thesis may be used to optimise switching functions and, in general, operate without regard to the target technology. This introductory chapter briefly discusses Reed-Muller (RM) expansions and considers the advantages and disadvantages of these forms of representation. Additionally, the contents of subsequent chapters of the document are previewed.

RM expansions provide an alternative means of representing switching functions. The RM expansion is based on the algebra of finite fields, or Galois fields. Galois fields are denoted $\operatorname{GF}(q)$ where $q$ is the number of elements in the field, and for $R M$ expansions, $q=2 . G F(2)$ is the smallest finite field whilst the set of real numbers is an example of a field with an infinite number of elements. The elements of $\mathrm{GF}(2)$ are 0 and 1 , and the algebraic operations defined in this field are modulo-2 addition and modulo2 multiplication. Modulo-2 addition is equivalent to the logical EXclusive-OR operation (EXOR) and modulo-2 multiplication corresponds to the logical AND operation. Hence, the RM representation of a switching function is an Exclusive-OR Sum-Of-Products (ESOP) of literals, where each literal may take the value 0 or $1[2,3,4]$.

There are certain advantages in representing switching functions as ESOP forms. Firstly, ESOP forms may provide more efficient representations than the traditional Boolean Sum-Of-Products (SOP) expressions [5, 6]. Arithmetic functions, which contain a substantial quantity of EXOR operations, are one type of function for which the ESOP representation may prove to be economical [7, 8]. The second advantage is that circuits which are implemented using AND and EXOR logic elements exhibit properties which are desirable in terms of testability [9, 10, 11]. These properties include reducing the size of the test sets which are required when testing for stuck-at faults and bridging faults. The inherent complexity of testing combinational circuits makes this advantage particularly interesting. As previously stated, the RM representation of a switching function is defined over GF(2), a special case of GF( $q$ ) or a finite field. Hence, an additional advantage lies in the possibility of extending techniques developed for RM expansions to operate in the other finite fields, in which multiple-valued switching functions are defined [6, 12, 13].

The disadvantages associated with the physical implementation of switching functions represented as ESOP forms are the main weaknesses in the case for promoting the use of synthesis tools based on ESOP forms. The EXOR gate is considered to be a complex gate and contains a greater number of transistors than the AND (NAND) and OR (NOR) gates normally employed to construct circuits. Although utilising pass-transistors can reduce the transistor count in an EXOR gate [14, 15], it is generally found that unless the ESOP representation comprises of significantly fewer product terms than the equivalent SOP form then the implementation of the ESOP form will be larger than the Boolean implementation [16]. The switching speed of an EXOR gate is longer than that of the basic Boolean logic gates. This factor also contributes to the case against using ESOP forms as a means of representing switching functions. However, developments in field programmable gate array (FPGA) technology, where the basic logic elements are blocks which can be programmed to perform simple logic functions, make it possible to realise EXOR gates which are comparable, in size and speed, to the basic logic gates [17, 18, 19]. These developments have to some extent overcome the practical problems which arise when implementing ESOP forms.

Following the initial definition of the RM expansion, interest in this particular means of representing switching functions has led to many diverse optimisation techniques and algorithms. These include various methods of presenting RM expansions and ESOP forms, each with inherent advantages and disadvantages. Additionally, techniques have been developed which generate canonical RM expansions from Boolean SOP forms. Exact and heuristic minimisation algorithms have evolved and may be utilised according to the types of ESOP forms which they optimise. That is, some techniques are suited to optimising only fixed polarity RM expansions whilst others optimise mixed polarity RM expansions, or the more general, unstructured ESOP forms.

It is interesting to note that developments in synthesis tools which utilise RM expansions and ESOP forms have to some extent followed the evolutionary route previously undertaken during the development of Boolean logic synthesis systems. Early Boolean logic synthesis tools were designed to realise efficient SOP forms. The optimised functions could then be implemented as two-level circuits using, for example, programmable logic arrays. Whilst this form of representation and implementation remains valuable, the increasing demands placed on the performance of logic systems means that two-level implementations are often unsatisfactory. This has led to the development of multi-level synthesis tools which rely on optimisation techniques such as factoring and decomposition. The resulting expansions, which may be represented as factored forms, can then be implemented using devices such as FPGAs. Following this trend, synthesis tools have been developed which derive multi-level implementations of RM expansions and ESOP forms. An approach which is currently generating considerable interest, and which may prove to be rewarding, is the development of 'mixed' synthesis systems. Here switching functions are partitioned and each subfunction represented either as an ESOP form or as a Boolean SOP form, thus exploiting the advantages of both types of representation [20].

Synthesis tools which exploit the advantages of ESOP representations of switching functions are becoming increasingly powerful. However, a significant amount of development work is required if the efficiency of the optimisation algorithms is to rival that of traditional Boolean logic synthesis
techniques.

This thesis presents logic synthesis and optimisation techniques which employ the RM representation. The research work undertaken and now detailed includes the development of several techniques and algorithms which may be employed to minimise switching functions. A basic logic synthesis system which incorporates both established techniques and these newly developed algorithms has been constructed. This fully automated package served as an aid in evaluating the efficiency of the algorithms and techniques developed throughout the duration of the project.

The theory presented in the second chapter commences with a basic review of the algebra of Galois fields and proceeds to detail RM expansions relative to Boolean SOP forms. The structure of fixed polarity and mixed polarity RM expansions and general ESOP forms are also described. The multi-level representation of switching functions using Binary Decision Diagrams (BDDs) is reviewed as a precursor to the research work presented in chapter 7. This is followed by a concise description of Reed-Muller Universal Logic Modules (RM-ULMs).

Chapter 3 commences with a review of the existing methods used to represent and generate fixed polarity RM expansions. Heuristic and exact minimisation techniques are also discussed. The chapter proceeds with the description of a heuristic algorithm for deriving optimal (sub-optimal) fixed polarity RM expansions. This technique employs a tabular means of representing both the switching functions and the fixed polarity ReedMuller expansions [1] and is an extension of a technique developed by Marinkovic and Tosic [21]. Results are presented which indicate the quality of the solutions produced by the new algorithm, taking into account the number of product terms in the final FPRM representation. Additionally, modified forms of the basic algorithm are suggested and results are presented which illustrate the effects of these modifications. The techniques are evaluated against established methods, using both randomly generated switching functions and a small set of benchmark functions.

Chapter 4 is dedicated to describing a method which has been developed to determine the optimum allocation of 'don't care' terms when deriving a
minimal RM expansion of predetermined fixed polarity from an initial incompletely specified Boolean function. The use of this technique in conjunction with the algorithm for determining 'good' fixed polarity RM expansions is also discussed, and results are presented which illustrate the most profitable use of the techniques.

The fifth chapter of this thesis reviews techniques for representing and generating mixed polarity RM expansions and the more general ESOP forms. Additionally, established optimisation methods are briefly discussed. An existing technique which is used to represent and generate fixed polarity RM forms is extended to enable the construction of KROnecker (KRO) expansions. This chapter is concluded with a description of an adapted form of this technique which allows the formation of KRO expansions from incompletely specified Boolean functions.

Although the derivation of RM expansions from Boolean functions expressed in minterm form is, in essence, a trivial task, the operation can consume substantial quantities of computer memory and processor time. The generation of RM representations from disjoint and non-disjoint Boolean SOP forms is reviewed in chapter 6. The tabular representation and conversion technique [1], reviewed in chapter 3, normally operates on an initial Boolean minterm representation of a switching function. The adaptation of this technique to allow the generation of RM forms from reduced Boolean SOP forms is discussed in the remainder of this chapter.

It has been previously stated in this thesis that two-level circuits do not always offer efficient means of implementing combinational logic functions. A practical solution may require a multi-level implementation which is derived using multi-level synthesis tools. Chapter 7 briefly reviews Boolean multi-level optimisation techniques and, in particular, considers the uses of Binary Decision Diagrams. A Reed-Muller Binary Decision Diagram (RMBDD) is introduced and the use of this form to represent fixed polarity RM expansions is demonstrated. Techniques for deriving minimal RMBDDs are also presented. The implementation of RMBDDs as both RM-ULM networks and multi-level circuits comprised of discrete logic gates is discussed in the latter part of this chapter.

Chapter 8 summarises the work detailed in this thesis and draws conclusions as to the advantages of performing logic synthesis using RM representations and ESOP forms. Additionally, areas suitable for future research are suggested.

## Chapter 2

## Theory and Definitions

Traditionally, switching circuits have been represented using operations defined in Boolean algebra. The Reed-Muller description of a switching function provides an alternative form of representation and is based on the algebraic operations defined over Galois Field(2) [2, 3, 4, 22]. The theory presented in this chapter briefly revises the algebraic operations defined over $\mathrm{GF}(2)$ and the relationships with Boolean algebra. The basic RM expansion is defined and derived from an initial Boolean sum-of-products form. Additionally, various types of RM expansions and exclusive-OR sum-of-products forms are reviewed. This includes detailed descriptions of fixed polarity Reed-Muller expansions, Kronecker expansions and Pseudo Kronecker expansions. The remainder of this chapter is dedicated to describing the structure of Binary Decision Diagrams and Reed-Muller Universal Logic Module networks.

### 2.1 Logic Functions

A logic function is a mapping

$$
f:\{0,1, \ldots, r-1\}^{n} \rightarrow\{0,1, \ldots, r-1\}
$$

where $n$ is the number of function variables, and $r$ is the cardinality of the set. Hence, each function variable may take $r$ different values.

A logic function is a switching function when $r=2$
i.e. $\quad f:\{0,1\}^{n} \rightarrow\{0,1\}$
where $n$ is the number of function variables and each function variable may take the value 0 or 1 .

An incompletely specified switching function is a function where, for one or more input conditions, the corresponding output states are undefined. This may be represented by the mapping

$$
f:\{0,1\}^{n} \rightarrow\{0,1, D\}
$$

where $D \in\{0,1\}$, denoting an undefined output state.

The functions which have been defined denote single-output logic and
switching functions. A multiple-output switching function may be represented by the mapping

$$
f:\{0,1\}^{n} \rightarrow\{0,1, D\}^{\infty}
$$

where $n$ is the number of function variables and $m$ is the number of output functions.

In order to clarify the terminology used in this thesis, it is necessary to state that reference to a function will imply a completely specified Boolean function. Other types of switching functions will be referred to explicitly.

### 2.2 Algebra of GF(2)

The operations defined over $G F(2)$ are modulo-2 addition and modulo-2 multiplication, and the elements defined in this field are the binary integers 0 and 1. Modulo-2 addition and modulo-2 multiplication are identical to the logical EXOR and logical AND operations, respectively. Hence, the operations defined for $\mathrm{GF}(2)$ algebra may be readily implemented using logic components. This is illustrated in Figure 2.1, which was presented by Green [3].

Throughout this thesis the symbol $\oplus$ denotes modulo-2 addition and the EXOR operation. The symbol + denotes logical addition and the Boolean inclusive-OR (OR) operation. The symbols $\odot$ and . denote modulo-2 multiplication and logical multiplication respectively. These two operations are equivalent to the Boolean AND operation and henceforth will be deemed to be identical. This operator may be omitted from all equations i.e. $x . y=$ $-x \circ y=x y$

In Figure 2.1 and the following equations, $x$ and $y$ are elements defined over GF(2) and may take the binary values 0 and 1.

The algebra of $\mathbf{G F}(2)$ obeys the law of closure, in addition to the associative, distributive and commutative laws. The identities $x \oplus 0=x$ and $x \odot 1=x$ are also satisfied.
Some additional properties exist due to the nature of $\mathrm{GF}(2)$ algebra $[2,23$ ] $x \oplus x=0 \quad x=-x$
hence, each element of $\mathrm{GF}(2)$ is its own additive inverse.
Further, $\quad x \circ x=x$
$\mathrm{GF}(2)$ algebra may be related to Boolean algebra,

$$
\begin{aligned}
& x \odot y=x \cdot y \\
& x \oplus y=x \cdot \bar{y}+\bar{x} \cdot y
\end{aligned}
$$

If $y=1$, then $x \oplus 1=x .(0)+\bar{x} .(1)=\bar{x}$

Additionally, employing De Morgans' theorem

$$
\begin{aligned}
x+y & =\overline{\bar{x} \cdot \bar{y}}=((x \oplus 1) \odot(y \oplus 1)) \oplus 1 \\
& =x \odot y \oplus x \oplus y \oplus 1 \oplus 1 \\
& =x \odot y \oplus x \oplus y
\end{aligned}
$$

Thus,

$$
\begin{aligned}
x . y & =x \odot y \\
x+y & =x \odot y \oplus x \oplus y \\
\bar{x} & =x \oplus 1
\end{aligned}
$$



Figure 2.1: Basic connectives of $G F(2)$ algebra and the equivalent logical operators.

### 2.3 Fixed Polarity Reed-Muller Expansions

The operations of $\mathrm{GF}(2)$ algebra have been defined in the preceding section. The structure of the Reed-Muller expansion, which employs the operations defined over $\mathrm{GF}(2)$, is now reviewed.

Any $n$ variable switching function may be represented in Boolean SOP form.

$$
\begin{align*}
f\left(x_{n}, x_{n-1}, \ldots, x_{1}\right) & =\sum_{i=0}^{20-1} d_{i} m_{i} \\
& =d_{0} \bar{x}_{n} \bar{x}_{n-1} \ldots \bar{x}_{2} \bar{x}_{1}+d_{1} \bar{x}_{n} \bar{x}_{n-1} \ldots \bar{x}_{2} x_{1}+d_{2} \bar{x}_{n} \bar{x}_{n-1} \ldots x_{2} \bar{x}_{1}+\ldots \ldots+d_{2 n-1} x_{n} x_{n-1} \ldots x_{2} x_{1} \tag{2.1}
\end{align*}
$$

$\Sigma$ denotes logical addition
$m_{i}$ denotes a minterm of the function
$d_{i} \in\{0,1\}$ is an operational domain coefficient
$i=0,1, \ldots, 2^{n}-1$
$x_{j}$ and $\bar{x}_{j}$ are literals of the function, in true and complemented forms respectively.
$j=1,2, \ldots, n$

A minterm $m_{i}$ is defined as a product of function variables and each minterm comprises of every function variable in either true or complemented form.

$$
m_{i}=\prod_{j=1}^{n} x_{j}^{i_{j}}
$$

where $i$ is the decimal representation of the binary $n$-tuple $\left\langle i_{n} i_{n-1} \ldots i_{2} i_{1}\right\rangle$, $i_{j} \in\{0,1\}$ and $x_{j}^{0}=\bar{x}_{j}, x_{j}^{1}=x_{j}$
e.g. for $n=3$, $m_{4}=m_{<100\rangle}=x_{3}^{1} x_{2}^{0} x_{1}^{0}=x_{3} \bar{x}_{2} \bar{x}_{1}$

The Boolean SOP form, in which each and every product term is a minterm, is described as the canonical disjunctive form (Equation (2.1)). It is possible to construct all $2^{2^{n}}$ possible Boolean functions of $n$ variables from this basic expansion by altering the values of the coefficients $d_{0}, \ldots, d_{2^{n}-1}$. Any Boolean function of $n$ variables may comprise of up to $2^{n}$ minterms. e.g. for $n=3$

$$
\begin{align*}
f\left(x_{3}, x_{2}, x_{1}\right)= & d_{0} \bar{x}_{3} \bar{x}_{2} \bar{x}_{1}+d_{1} \bar{x}_{3} \bar{x}_{2} x_{1}+d_{2} \bar{x}_{3} x_{2} \bar{x}_{1}+d_{3} \bar{x}_{3} x_{2} x_{1} \\
& +d_{4} x_{3} \bar{x}_{2} \bar{x}_{1}+d_{5} x_{3} \bar{x}_{2} x_{1}+d_{6} x_{3} x_{2} \bar{x}_{1}+d_{3} x_{3} x_{2} x_{1} \tag{2.2}
\end{align*}
$$

The coefficients $d_{0}, \ldots, d_{2^{n}-1}$ of the canonic Boolean SOP form correspond
directly to the output of the truth table representation and hence, to the operation of the function. The forms of representation illustrated in Equations (2.1) and (2.2) can, therefore, be termed operational domain descriptions of switching functions.

The minterms of a Boolean function are mutually exclusive (or disjoint), i.e. $m_{i} m_{k}=0$ for all $i \neq k, i, k=0,1, \ldots, 2^{n}-1$. This property may be exploited making it possible to replace the inclusive-OR operator with the exclusiveOR operator without altering the operation of the expansion. This forms an exclusive-OR (or ring) sum-of-products expansion of the function [24, 25].

$$
\begin{align*}
f\left(x_{n}, x_{n-1}, \ldots, x_{1}\right) & =\oplus \sum_{i=0}^{2^{n}-1} d_{i} m_{i} \\
& =d_{0} \bar{x}_{n} \bar{x}_{n-1} \ldots \bar{x}_{2} \bar{x}_{1} \oplus d_{1} \bar{x}_{n} \bar{x}_{n-1} \ldots \bar{x}_{2} x_{1} \oplus d_{2} \bar{x}_{n} \bar{x}_{n-1} \ldots x_{2} \bar{x}_{1} \oplus \ldots \ldots \oplus d_{2^{n}-1} x_{n} x_{n-1} \ldots x_{2} x_{1} \tag{2.3}
\end{align*}
$$

$\oplus \Sigma$ denotes the ring sum (modulo- 2 addition), $m_{i}, d_{i}, i, x_{j}, \bar{x}_{j}$ and $j$ are as defined for Equation (2.1). The expansion of Equation (2.3) is the ESOP representation of the Boolean function described by Equation (2.1).

The Reed-Muller expansion is defined as the complement-free ring sum-ofproducts expression of a switching function. This may be derived from the expression detailed in Equation (2.3) by employing the substitution $\bar{x}_{j}=x_{j} \oplus 1$ for $j=1,2, \ldots, n$.
Hence, from Equation (2.3)

$$
\begin{aligned}
f\left(x_{n}, x_{n-1}, \ldots, x_{1}\right)= & d_{0}\left(x_{n} \oplus 1\right)\left(x_{n-1} \oplus 1\right) \ldots\left(x_{2} \oplus 1\right)\left(x_{1} \oplus 1\right) \oplus d_{1}\left(x_{n} \oplus 1\right)\left(x_{n-1} \oplus 1\right) \ldots\left(x_{2} \oplus 1\right) x_{1} \\
& \oplus d_{2}\left(x_{n} \oplus 1\right)\left(x_{n-1} \oplus 1\right) \ldots x_{2}\left(x_{1} \oplus 1\right) \oplus \ldots \ldots \oplus d_{2 n-1} x_{n} x_{n-1} \ldots x_{2} x_{1} \\
= & d_{0} \oplus\left(d_{0} \oplus d_{1}\right) x_{1} \oplus\left(d_{0} \oplus d_{2}\right) x_{2} \oplus\left(d_{0} \oplus d_{1} \oplus d_{2} \oplus d_{3}\right) x_{2} x_{1} \\
& \oplus\left(d_{0} \oplus d_{1}\right) x_{3} \oplus \ldots \ldots \oplus\left(d_{0} \oplus d_{1} \oplus d_{2} \oplus \ldots \oplus d_{2 n-1}\right) x_{n} x_{n-1} \ldots x_{2} x_{1}
\end{aligned}
$$

The equivalent RM expansion is defined as

$$
\begin{align*}
f\left(x_{n}, x_{n-1}, \ldots, x_{1}\right) & =\sum_{i=0}^{20-1} a_{i} \pi_{i} \\
& =a_{0} \oplus a_{1} x_{1} \oplus a_{2} x_{2} \oplus a_{3} x_{2} x_{1} \oplus \ldots \ldots \oplus a_{2-1} x_{n} x_{n-1} \ldots x_{2} x_{1} \tag{2.4}
\end{align*}
$$

$\pi_{f}$ is a piterm of the expansion
$a_{i} \in\{0,1\}$ is a functional domain coefficient
$i=0,1, \ldots, 2^{n}-1$
$x_{j}$ is a literal of the RM expansion (and of the equivalent Boolean function) and is present only in true form.
$j=1,2, \ldots, n$

A piterm, $\pi_{i}$, of a $R M$ expansion is defined as a product of expansion variables,

$$
\pi_{1}=\prod_{j=1}^{n} x_{j}^{x_{j}}
$$

where $i$ is the decimal representation of the binary $n$-tuple $\left\langle i_{n} i_{n-1} \ldots i_{2} i_{1}>\right.$, $i_{j} \in\{0,1\}$ and $x_{j}^{0}=1, x_{j}^{1}=x_{j}$
e.g. for $n=3$,
$\pi_{4}=\pi_{<100\rangle}=x_{3}^{1} x_{2}^{0} x_{1}^{0}=x_{3}$

The RM expansion of Equation (2.4) is a canonic form which uniquely represents the initial Boolean function. It is possible to derive all $2^{2^{n}} \mathrm{RM}$ expansions of $n$ variables from this basic expression simply by altering the values of the coefficients $a_{0}, \ldots, a_{2^{n}-1}$. A RM expansion of $n$ variables may comprise of up to $2^{n}$ piterms.
e.g. for $n=3$

$$
f\left(x_{3}, x_{2}, x_{1}\right)=a_{0} \oplus a_{1} x_{1} \oplus a_{2} x_{2} \oplus a_{3} x_{2} x_{1} \oplus a_{4} x_{3} \oplus a_{3} x_{3} x_{1} \oplus a_{6} x_{3} x_{2} \oplus a_{1} x_{3} x_{2} x_{1}
$$

The coefficients $d_{i}\left(i=0,1, \ldots, 2^{n}-1\right)$ of a Boolean function are termed the operational domain coefficients and correspond to the output of the truth table representation. They directly represent the operation of the function. The transformation from the Boolean domain to the Reed-Muller domain alters the significance of these coefficients. Hence, the coefficients of the RM expansion, $a_{1}\left(i=0,1, \ldots, 2^{n}-1\right)$, are termed the functional domain coefficients and no longer directly correspond to the output of the truth table representation of the Boolean function. The relationship between the operational domain and functional domain coefficients is illustrated for the particular case of $n=3$, i.e. a switching function of 3 variables. However,
this relationship can be extended to switching functions of any number of variables.

Boolean function ( $n=3$ )

$$
\begin{aligned}
f\left(x_{3} x_{2}, x_{1}\right)= & d_{0} \bar{x}_{3} \bar{x}_{2} \bar{x}_{1}+d_{1} \bar{x}_{3} \bar{x}_{2} x_{1}+d_{2} \bar{x}_{3} x_{2} \bar{x}_{1}+d_{3} \bar{x}_{3} x_{2} x_{1} \\
& +d_{4} x_{3} \bar{x}_{2} \bar{x}_{1}+d_{5} x_{3} \bar{x}_{2} x_{1}+d_{6} x_{3} x_{2} \bar{x}_{1}+d_{1} x_{3} x_{2} x_{1}
\end{aligned}
$$

As the minterms are disjoint the OR operator may be directly replaced by the EXOR operator,

$$
\begin{aligned}
f\left(x_{3} x_{2}, x_{1}\right)= & d_{0} \bar{x}_{3} \bar{x}_{2} \bar{x}_{1} \oplus d_{1} \bar{x}_{3} \bar{x}_{2} x_{1} \oplus d_{2} \bar{x}_{3} x_{2} \bar{x}_{1} \oplus d_{3} \bar{x}_{3} x_{2} x_{1} \\
& \oplus d_{4} x_{3} \bar{x}_{2} \bar{x}_{1} \oplus d_{5} x_{3} \bar{x}_{2} x_{1} \oplus d_{6} x_{3} x_{2} \bar{x}_{1} \oplus d_{1} x_{3} x_{2} x_{1}
\end{aligned}
$$

Employ the substitution $\bar{x}_{j}=x_{j} \oplus 1(j=1,2, \ldots, n)$

$$
\begin{aligned}
f\left(x_{3} x_{2}, x_{1}\right)= & d_{0}\left(x_{3} \oplus 1\right)\left(x_{2} \oplus 1\right)\left(x_{1} \oplus 1\right) \oplus d_{1}\left(x_{3} \oplus 1\right)\left(x_{2} \oplus 1\right) x_{1} \\
& \oplus d_{2}\left(x_{3} \oplus 1\right) x_{2}\left(x_{1} \oplus 1\right) \oplus d_{3}\left(x_{3} \oplus 1\right) x_{2} x_{1} \oplus d_{4} x_{3}\left(x_{2} \oplus 1\right)\left(x_{1} \oplus 1\right) \\
& \oplus d_{5} x_{3}\left(x_{2} \oplus 1\right) x_{1} \oplus d_{6} x_{3} x_{2}\left(x_{1} \oplus 1\right) \oplus d_{1} x_{3} x_{2} x_{1} \\
= & d_{0}\left(x_{3} x_{2} x_{1} \oplus x_{3} x_{2} \oplus x_{3} x_{1} \oplus x_{3} \oplus x_{2} x_{1} \oplus x_{2} \oplus x_{1} \oplus 1\right) \\
& \oplus d_{1}\left(x_{3} x_{2} x_{1} \oplus x_{3} x_{1} \oplus x_{2} x_{1} \oplus x_{1}\right) \oplus d_{2}\left(x_{3} x_{2} x_{1} \oplus x_{3} x_{2} \oplus x_{2} x_{1} \oplus x_{2}\right) \\
& \oplus d_{3}\left(x_{3} x_{2} x_{1} \oplus x_{2} x_{1}\right) \oplus d_{4}\left(x_{3} x_{2} x_{1} \oplus x_{3} x_{2} \oplus x_{3} x_{1} \oplus x_{3}\right) \\
& \oplus d_{5}\left(x_{3} x_{2} x_{1} \oplus x_{3} x_{1}\right) \oplus d_{6}\left(x_{3} x_{2} x_{1} \oplus x_{3} x_{2}\right) \oplus d_{1} x_{3} x_{2} x_{1} \\
= & d_{0} \oplus\left(d_{0} \oplus d_{1}\right) x_{1} \oplus\left(d_{0} \oplus d_{2}\right) x_{2} \oplus\left(d_{0} \oplus d_{1} \oplus d_{2} \oplus d_{3}\right) x_{2} x_{1} \\
& \oplus\left(d_{0} \oplus d_{4}\right) x_{3} \oplus\left(d_{0} \oplus d_{1} \oplus d_{4} \oplus d_{5}\right) x_{3} x_{1} \oplus\left(d_{0} \oplus d_{2} \oplus d_{4} \oplus d_{6}\right) x_{3} x_{2} \\
& \oplus\left(d_{0} \oplus d_{1} \oplus d_{2} \oplus d_{3} \oplus d_{4} \oplus d_{5} \oplus d_{6} \oplus d_{1}\right) x_{3} x_{2} x_{1} \\
= & a_{0} \oplus a_{1} x_{1} \oplus a_{2} x_{2} \oplus a_{3} x_{2} x_{1} \oplus a_{4} x_{3} \oplus a_{3} x_{3} x_{1} \oplus a_{6} x_{3} x_{2} \oplus a_{1} x_{3} x_{2} x_{1}
\end{aligned}
$$

Reed-Muller expansion ( $n=3$ )

$$
f\left(x_{3}, x_{2}, x_{1}\right)=a_{0} \oplus a_{1} x_{1} \oplus a_{2} x_{2} \oplus a_{3} x_{2} x_{1} \oplus a_{4} x_{3} \oplus a_{5} x_{3} x_{1} \oplus a_{6} x_{3} x_{2} \oplus a_{7} x_{3} x_{2} x_{1}
$$

The functional domain coefficients may be related to the operational domain coefficients in the following manner.

$$
\begin{aligned}
& a_{0}=d_{0} \\
& a_{1}=d_{0} \oplus d_{1} \\
& a_{2}=d_{0} \oplus d_{2}
\end{aligned}
$$

$$
\begin{aligned}
& a_{3}=d_{0} \oplus d_{1} \oplus d_{2} \oplus d_{3} \\
& a_{4}=d_{0} \oplus d_{4} \\
& a_{5}=d_{0} \oplus d_{1} \oplus d_{4} \oplus d_{5} \\
& a_{6}=d_{0} \oplus d_{2} \oplus d_{4} \oplus d_{6} \\
& a_{7}=d_{0} \oplus d_{1} \oplus d_{2} \oplus d_{3} \oplus d_{4} \oplus d_{5} \oplus d_{6} \oplus d_{7}
\end{aligned}
$$

Alternatively,

$$
\begin{aligned}
& d_{0}=a_{0} \\
& d_{1}=a_{0} \oplus a_{1} \\
& d_{2}=a_{0} \oplus a_{2} \\
& d_{3}=a_{0} \oplus a_{1} \oplus a_{2} \oplus a_{3} \\
& d_{4}=a_{0} \oplus a_{4} \\
& d_{5}=a_{0} \oplus a_{1} \oplus a_{4} \oplus a_{5} \\
& d_{6}=a_{0} \oplus a_{2} \oplus a_{4} \oplus a_{6} \\
& d_{7}=a_{0} \oplus a_{1} \oplus a_{2} \oplus a_{3} \oplus a_{4} \oplus a_{5} \oplus a_{6} \oplus a_{7}
\end{aligned}
$$

The RM expansion (Equation (2.4)) is the basic canonical ESOP expression and all expansion variables are present in true form throughout the expression. It is possible to derive a further $2^{n}-1$ canonical ESOP expansions from this basic form where each new expansion has some combination of variables present in complemented form throughout the expression. This is realised by utilising the substitution $x_{j}=\bar{x}_{j} \oplus 1$. The $2^{n}$ canonical forms (including the original RM expansion) are termed fixed polarity Reed-Muller (FPRM) expansions and have the general form

$$
\begin{align*}
f\left(x_{n}, x_{n-1}, \ldots, x_{1}\right) & =\oplus \sum_{i=0}^{2^{n}-1} b_{1} p_{i} \\
& =b_{0} \oplus b_{1} \dot{x}_{1} \oplus b_{2} \dot{x}_{2} \oplus b_{3} \dot{x}_{2} \dot{x}_{1} \oplus \ldots \ldots b_{2^{n}-1} \dot{x}_{n} \dot{x}_{n-1} \ldots \dot{x}_{2} \dot{x}_{1} \tag{2.5}
\end{align*}
$$

$\rho_{i}$ is a product term of the expansion
$b_{i} \in\{0,1\}$ is a functional domain coefficient
$i=0,1, \ldots, 2^{n_{-1}}$
$\dot{x}_{j}=x_{j}$ or $\bar{x}_{j}$, that is, a FPRM expansion may contain $x_{j}$ or $\bar{x}_{j}$ but not both. $j=1,2, \ldots, n$

The product terms of the FPRM expansion may be defined in a manner similar to that used to define the piterms of the original RM expansion.

$$
P_{l}=\prod_{j=1}^{n} \dot{x}_{j}^{i_{j}}
$$

where $i$ is the decimal representation of the binary $n$-tuple $\left\langle i_{n} i_{n-1} \ldots i_{2} i_{1}\right\rangle$, $i_{j} \in\{0,1\}$ and $\dot{x}_{j}^{0}=1, \dot{x}_{j}^{1}=\dot{x}_{j}$ $\dot{x}_{j}=x_{j}$ or $\bar{x}_{j}$

The coefficients, $b_{i}$, of each FPRM expansion are related to the coefficients, $a_{1}\left(i=0,1, \ldots, 2^{n}-1\right)$, of the RM expansion (Equation (2.4)). The FPRM expansion where all variables are present in true form is equivalent to the RM expansion. Hence, the coefficients of each expression may be directly equated with one another, i.e. $a_{1}=b_{i}$ and $b_{i}=a_{1}$ for all $i$. However, in order to relate the $b_{i}$ coefficients of the remaining FPRM expansions to the $a_{1}$ coefficients of the RM expansion it is necessary to expand each FPRM expansion, employing the substitution $\bar{x}_{j}=x_{j} \oplus 1(j \in\{1,2, \ldots, n\})$.
The relationship is illustrated for the particular case of the FPRM expansion of 3 variables, where $x_{3}$ and $x_{1}$ are present in complemented form and $x_{2}$ is present in true form.
FPRM expansion ( $n=3$ )

$$
\begin{equation*}
f\left(x_{3}, x_{2}, x_{1}\right)=b_{0} \oplus b_{1} \bar{x}_{1} \oplus b_{2} x_{2} \oplus b_{3} x_{2} \bar{x}_{1} \oplus b_{4} \bar{x}_{3} \oplus b_{5} \bar{x}_{3} \bar{x}_{1} \oplus b_{6} \bar{x}_{3} x_{2} \oplus b_{7} \bar{x}_{3} x_{2} \bar{x}_{1} \tag{2.6}
\end{equation*}
$$

Employ the substitutions $\bar{x}_{3}=x_{3} \oplus 1$ and $\bar{x}_{1}=x_{1} \oplus 1$,

$$
\begin{align*}
f\left(x_{3}, x_{2}, x_{1}\right)= & b_{0} \oplus b_{1}\left(x_{1} \oplus 1\right) \oplus b_{2} x_{2} \oplus b_{3} x_{2}\left(x_{1} \oplus 1\right) \oplus b_{4}\left(x_{3} \oplus 1\right) \\
& \oplus b_{5}\left(x_{3} \oplus 1\right)\left(x_{1} \oplus 1\right) \oplus b_{6}\left(x_{3} \oplus 1\right) x_{2} \oplus b_{7}\left(x_{3} \oplus 1\right) x_{2}\left(x_{1} \oplus 1\right) \\
= & b_{0} \oplus b_{1} x_{1} \oplus b_{1} \oplus b_{2} x_{2} \oplus b_{3} x_{2} x_{1} \oplus b_{3} x_{2} \oplus b_{4} x_{3} \oplus b_{4} \oplus b_{3} x_{3} x_{1} \oplus b_{5} x_{3} \\
& \oplus b_{5} x_{1} \oplus b_{5} \oplus b_{6} x_{3} x_{2} \oplus b_{6} x_{2} \oplus b_{7} x_{3} x_{2} x_{1} \oplus b_{7} x_{3} x_{2} \oplus b_{7} x_{2} x_{1} \oplus b_{7} x_{2} \\
= & \left(b_{0} \oplus b_{1} \oplus b_{4} \oplus b_{5}\right) \oplus\left(b_{1} \oplus b_{5}\right) x_{1} \oplus\left(b_{2} \oplus b_{3} \oplus b_{6} \oplus b_{7}\right) x_{2} \\
& \oplus\left(b_{3} \oplus b_{7}\right) x_{2} x_{1} \oplus\left(b_{4} \oplus b_{5}\right) x_{3} \oplus b_{5} x_{3} x_{1} \oplus\left(b_{6} \oplus b_{7}\right) x_{3} x_{2} \\
& \oplus b_{7} x_{3} x_{2} x_{1} \tag{2.7}
\end{align*}
$$

RM expansion ( $n=3$ )

$$
\begin{equation*}
f\left(x_{3}, x_{2}, x_{1}\right)=a_{0} \oplus a_{1} x_{1} \oplus a_{2} x_{2} \oplus a_{3} x_{2} x_{1} \oplus a_{4} x_{3} \oplus a_{5} x_{3} x_{1} \oplus a_{6} x_{3} x_{2} \oplus a_{1} x_{3} x_{2} x_{1} \tag{2.8}
\end{equation*}
$$

If the expansions of Equations (2.7) and (2.8) are compared then the coefficients of the RM expansion (Equation (2.8)) may be related to the coefficients of the FPRM expansion (Equation (2.6)) in the following manner.

$$
\begin{aligned}
& a_{0}=b_{0} \oplus b_{1} \oplus b_{4} \oplus b_{5} \\
& a_{1}=b_{1} \oplus b_{5} \\
& a_{2}=b_{2} \oplus b_{3} \oplus b_{6} \oplus b_{7} \\
& a_{3}=b_{3} \oplus b_{7} \\
& a_{4}=b_{4} \oplus b_{5} \\
& a_{5}=b_{5} \\
& a_{6}=b_{6} \oplus b_{7} \\
& a_{7}=b_{7}
\end{aligned}
$$

Alternatively,

$$
\begin{aligned}
& b_{0}=a_{0} \oplus a_{1} \oplus a_{4} \oplus a_{5} \\
& b_{1}=a_{1} \oplus a_{5} \\
& b_{2}=a_{2} \oplus a_{3} \oplus a_{6} \oplus a_{7} \\
& b_{3}=a_{3} \oplus a_{7} \\
& b_{4}=a_{4} \oplus a_{5} \\
& b_{5}=a_{5} \\
& b_{6}=a_{6} \oplus a_{7} \\
& b_{7}=a_{7}
\end{aligned}
$$

As the coefficients $b_{1}$ are related to the coefficients $a_{1}$ it is also possible to relate the $b_{1}$ coefficients to the operational domain coefficients $d_{1}$ of the Boolean SOP expansion (Equation (2.1)).

A FPRM expansion $f\left(x_{n}, x_{n-1}, \ldots, x_{1}\right)$ may be identified by means of a polarity number $p, 0 \leq p \leq 2^{n}-1$. Hence, the polarity $p$ FPRM expansion may be denoted $f_{p}\left(x_{n}, x_{n-1}, \ldots, x_{1}\right)$. This number, $p$, indicates the state of each expansion variable throughout the expression, that is, which variables are present in true form and which in complemented form. The polarity number, $p$, is the decimal equivalent of the binary $n$-tuple $\left\langle p_{n} p_{n-1} \ldots p_{1}\right\rangle$, where $p_{j}$ is replaced by 0 if $x_{j}$ is present throughout the FPRM expansion. If $\bar{x}_{j}$ is present throughout the FPRM expansion then $p_{j}$ is replaced by 1 . If this notation is applied to the basic RM expansion, where all variables are present in true form then $p_{j}=0$ for $j=1,2, \ldots, n$, hence, $p=0$ and this expression is the polarity 0 FPRM expansion. This is also known as the positive polarity RM (PPRM) expansion. The polarity of the FPRM expansion where all variables are present in complemented form can be determined by
setting $p_{j}=1$ for $j=1,2, \ldots, n$, resulting in $p=2^{n}-1$. The polarity $2^{n}-1$ FPRM expansion is also called the negative polarity RM expansion.

The following expressions further illustrate the use of polarity numbers to identify FPRM expansions.
e.g. for $n=3$,

$$
\begin{array}{ll}
p=0=<000\rangle & p_{3}=0 \rightarrow x_{3} \\
p_{2}=0 \rightarrow x_{2} \\
& p_{1}=0 \rightarrow x_{1}
\end{array}
$$

Polarity 0 FPRM expansion (also termed the positive polarity RM expansion).

$$
\begin{equation*}
f_{0}\left(x_{3}, x_{2}, x_{1}\right)=a_{0} \oplus a_{1} x_{1} \oplus a_{2} x_{2} \oplus a_{3} x_{2} x_{1} \oplus a_{4} x_{3} \oplus a_{3} x_{3} x_{1} \oplus a_{6} x_{3} x_{2} \oplus a_{1} x_{3} x_{2} x_{1} \tag{2.9}
\end{equation*}
$$

$$
p=3=<011\rangle \quad \begin{aligned}
& p_{3}=0 \rightarrow x_{3} \\
& p_{2}=1 \rightarrow \bar{x}_{2} \\
& p_{1}=1 \rightarrow \bar{x}_{1}
\end{aligned}
$$

Polarity 3 FPRM expansion.

$$
\begin{align*}
f_{3}\left(x_{3}, x_{2}, x_{1}\right)= & \left(a_{0} \oplus a_{1} \oplus a_{2} \oplus a_{3}\right) \oplus\left(a_{1} \oplus a_{3}\right) \bar{x}_{1} \oplus\left(a_{2} \oplus a_{3}\right) \bar{x}_{2} \oplus a_{3} \bar{x}_{2} \bar{x}_{1} \\
& \oplus\left(a_{4} \oplus a_{5} \oplus a_{6} \oplus a_{7}\right) x_{3} \oplus\left(a_{3} \oplus a_{7}\right) x_{3} \bar{x}_{1} \oplus\left(a_{6} \oplus a_{7}\right) x_{3} \bar{x}_{2} \\
& \oplus a_{7} x_{3} \bar{x}_{2} \bar{x}_{1} \tag{2.10}
\end{align*}
$$

$$
\begin{aligned}
& p=7=<111>
\end{aligned} \begin{aligned}
& p_{3}=1 \rightarrow \bar{x}_{3} \\
& p_{2}=1 \rightarrow \bar{x}_{2} \\
& p_{1}=1 \rightarrow \bar{x}_{1}
\end{aligned}
$$

Polarity 7 FPRM expansion (also termed the negative polarity RM expansion).

$$
\begin{align*}
f_{7}\left(x_{3}, x_{2}, x_{1}\right)= & \left(a_{0} \oplus a_{1} \oplus a_{2} \oplus a_{3} \oplus a_{4} \oplus a_{5} \oplus a_{6} \oplus a_{7}\right) \\
& \oplus\left(a_{1} \oplus a_{3} \oplus a_{5} \oplus a_{7}\right) \bar{x}_{1} \oplus\left(a_{2} \oplus a_{3} \oplus a_{6} \oplus a_{7}\right) \bar{x}_{2} \\
& \oplus\left(a_{3} \oplus a_{7}\right) \bar{x}_{2} \bar{x}_{1} \oplus\left(a_{4} \oplus a_{3} \oplus a_{6} \oplus a_{7}\right) \bar{x}_{3} \oplus \\
& \oplus\left(a_{5} \oplus a_{7}\right) \bar{x}_{3} \bar{x}_{1} \oplus\left(a_{6} \oplus a_{7}\right) \bar{x}_{3} \bar{x}_{2} \oplus a_{7} \bar{x}_{3} \bar{x}_{2} \bar{x}_{1} \tag{2.11}
\end{align*}
$$

2.4 Shannon Expansion Theorem and Exclusive-OR Sum-of-Products Forms The Shannon expansion theorem [26] forms a series expansion of a
switching function, $f\left(x_{n}, x_{n-1}, \ldots, x_{1}\right)$, and has the general form

$$
\begin{equation*}
f\left(x_{n}, x_{n-1}, \ldots, x_{1}\right)=\bar{x}_{j} f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{2} x_{1}\right)+x_{j} f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{2}, x_{1}\right) \tag{2.12}
\end{equation*}
$$

where the coefficients of $x_{j}$ and $\bar{x}_{f}$ namely $f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{2}, x_{1}\right)$ and $f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{2}, x_{1}\right)(j=1,2, \ldots, n)$ are subfunctions of $f\left(x_{n}, x_{n-1}\right.$, $\left.\ldots, x_{1}\right)$, and are themselves switching functions of ( $n-1$ ) variables. These functions of ( $n-1$ ) variables may be expanded about any variable $x_{k}$ ( $k=$ $1,2, \ldots, n \quad k \neq j)$. This operation may be used in a recursive manner until the original function has been expanded about all $n$ variables. The switching function is then represented by the expansion

$$
\begin{align*}
f\left(x_{n} x_{n-1}, \ldots, x_{1}\right)= & \bar{x}_{n} \bar{x}_{n-1} \ldots \bar{x}_{2} \bar{x}_{1} f(0,0, \ldots, 0,0)+\bar{x}_{n} \bar{x}_{n-1} \ldots \bar{x}_{2} x_{1} f(0,0, \ldots, 0,1) \\
& +\bar{x}_{n} \bar{x}_{n-1} \ldots \bar{x}_{2} \bar{x}_{1} f(0,0, \ldots, 1,0)+\ldots \ldots+x_{n} x_{n-1} \ldots x_{2} x_{1} f(1,1, \ldots, 1,1) \tag{2.13}
\end{align*}
$$

Comparing Equations (2.1) and (2.13) it can be seen that the $2^{n}$ coefficients, $f(0,0, \ldots, 0,0), f(0,0, \ldots, 0,1), \ldots \ldots, f(1,1, \ldots, 1,1)$, can be equated with the operational domain coefficients $d_{i}\left(i=0,1, \ldots, 2^{n}-1\right)$.
Hence,

$$
\begin{aligned}
d_{0} & =f(0,0, \ldots, 0,0) \\
d_{1} & =f(0,0, \ldots, 0,1) \\
d_{2} & =f(0,0, \ldots, 1,0) \\
d_{3} & =f(0,0, \ldots, 1,1) \\
\cdot & \\
\cdot & \\
d_{2^{n}-2} & =f(1,1, \ldots, 1,0) \\
d_{2^{n}-2} & =f(1,1, \ldots, 1,1)
\end{aligned}
$$

The expansion of Equation (2.12) represents the canonic disjunctive form of a switching function. It is, therefore, possible to replace the inclusiveOR operator with the exclusive-OR operator without altering the validity of the expression. The resulting expansion is

$$
\begin{equation*}
f\left(x_{n}, x_{n-1}, \ldots, x_{1}\right)=\bar{x}_{j} f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{2} x_{1}\right) \oplus x_{j} f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{2}, x_{1}\right) \tag{2.14}
\end{equation*}
$$

Employing the substitutions $\bar{x}_{j}=x_{j} \oplus 1$ and $x_{j}=\bar{x}_{j} \oplus 1$ it is possible to
derive two further expansions

$$
\begin{align*}
f\left(x_{n}, x_{n-1}, \ldots, x_{1}\right)= & f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{2}, x_{1}\right) \\
& \bullet x_{j}\left[f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{2} x_{1}\right) \bullet f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{2}, x_{1}\right)\right] \tag{2.15}
\end{align*}
$$

$$
\begin{align*}
f\left(x_{n}, x_{n-1}, \ldots, x_{1}\right)= & f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{2}, x_{1}\right) \\
& \bullet \bar{x}_{j}\left[f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{2} x_{1}\right) \oplus f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{2}, x_{1}\right)\right] \tag{2.16}
\end{align*}
$$

The expansion given in Equation (2.15) may be used iteratively to construct the polarity 0 FPRM expansion of $n$ variables. Each of the remaining $2^{n}-1$ FPRM expansions of a $n$ variable switching function may be constructed by expanding the appropriate combination of the expressions represented in Equations (2.15) and (2.16).
e.g. for $n=3$ and $p=4$, variable $x_{3}$ is present in complemented form,

$$
\begin{aligned}
f\left(x_{3}, x_{2}, x_{1}\right) & =\bar{x}_{3} f\left(0, x_{2}, x_{1}\right) \oplus x_{3} f\left(1, x_{2}, x_{1}\right) \\
& =f\left(1, x_{2}, x_{1}\right) \oplus \bar{x}_{3}\left[f\left(0, x_{2}, x_{1}\right) \oplus f\left(1, x_{2}, x_{1}\right)\right]
\end{aligned}
$$

Variable $x_{2}$ is present in true form,

$$
\begin{aligned}
f\left(0, x_{2}, x_{1}\right) & =\bar{x}_{2} f\left(0,0, x_{1}\right) \oplus x_{2} f\left(0,1, x_{1}\right) \\
& =f\left(0,0, x_{1}\right) \oplus x_{2}\left[f\left(0,0, x_{1}\right) \oplus f\left(0,1, x_{1}\right)\right] \\
f\left(1, x_{2}, x_{1}\right) & =\bar{x}_{2} f\left(1,0, x_{1}\right) \oplus x_{2} f\left(1,1, x_{1}\right) \\
& =f\left(1,0, x_{1}\right) \oplus x_{2}\left[f\left(1,0, x_{1}\right) \oplus f\left(1,1, x_{1}\right)\right]
\end{aligned}
$$

Variable $x_{1}$ is present in true form,

$$
\begin{aligned}
f\left(0,0, x_{1}\right) & =\bar{x}_{1} f(0,0,0) \oplus x_{1} f(0,0,1) \\
& =f(0,0,0) \oplus x_{1}[f(0,0,0) \oplus f(0,0,1)] \\
f\left(0,1, x_{1}\right) & =\bar{x}_{1} f(0,1,0) \oplus x_{1} f(0,1,1) \\
& =f(0,1,0) \oplus x_{1}[f(0,1,0) \oplus f(0,1,1)]
\end{aligned}
$$

$$
\begin{aligned}
f\left(1,0, x_{1}\right) & =\bar{x}_{1} f(1,0,0) \oplus x_{1} f(1,0,1) \\
& =f(1,0,0) \oplus x_{1}[f(1,0,0) \oplus f(1,0,1)] \\
f\left(1,1, x_{1}\right) & =\bar{x}_{1} f(1,1,0) \oplus x_{1} f(1,1,1) \\
& =f(1,1,0) \oplus x_{1}[f(1,1,0) \oplus f(1,1,1)]
\end{aligned}
$$

The polarity 4 FPRM expansion may be formed by substituting the appropriate expressions for each subfunction. Thus,

$$
\begin{aligned}
f_{4}\left(x_{3}, x_{2}, x_{1}\right)= & f\left(1, x_{2}, x_{1}\right) \oplus \bar{x}_{3}\left[f\left(0, x_{2}, x_{1}\right) \oplus f\left(1, x_{2}, x_{1}\right)\right] \\
= & f\left(1,0, x_{1}\right) \oplus x_{2}\left[f\left(1,0, x_{1}\right) \oplus f\left(1,1, x_{1}\right)\right] \\
& \oplus \bar{x}_{3}\left[f\left(0,0, x_{1}\right) \oplus x_{2}\left[f\left(0,0, x_{1}\right) \oplus f\left(0,1, x_{1}\right)\right]\right. \\
& \left.\oplus f\left(1,0, x_{1}\right) \oplus x_{2}\left[f\left(1,0, x_{1}\right) \oplus f\left(1,1, x_{1}\right)\right]\right] \\
= & f(1,0,0) \oplus x_{1}[f(1,0,0) \oplus f(1,0,1)] \\
& \oplus x_{2}\left[f(1,0,0) \oplus x_{1}[f(1,0,0) \oplus f(1,0,1)]\right. \\
& \left.\oplus f(1,1,0) \oplus x_{1}[f(1,1,0) \oplus f(1,1,1)]\right] \\
& \oplus \bar{x}_{3}\left[f(0,0,0) \oplus x_{1}[f(0,0,0) \oplus f(0,0,1)]\right. \\
& \oplus x_{2}\left[f(0,0,0) \oplus x_{1}[f(0,0,0) \oplus f(0,0,1)]\right. \\
& \left.\oplus f(0,1,0) \oplus x_{1}[f(0,1,0) \oplus f(0,1,1)]\right] \\
& \oplus f(1,0,0) \oplus x_{1}[f(1,0,0) \oplus f(1,0,1)] \\
& \oplus x_{2}\left[f(1,0,0) \oplus x_{1}[f(1,0,0) \oplus f(1,0,1)]\right. \\
& \left.\left.\oplus f(1,1,0) \oplus x_{1}[f(1,1,0) \oplus f(1,1,1)]\right]\right]
\end{aligned}
$$

Rearrange

$$
\begin{aligned}
f_{4}\left(x_{3}, x_{2}, x_{1}\right)= & f(1,0,0) \oplus x_{1}[f(1,0,0) \oplus f(1,0,1)] \\
& \oplus x_{2}[f(1,0,0) \oplus f(1,1,0)] \\
& \oplus x_{2} x_{1}[f(1,0,0) \oplus f(1,0,1) \oplus f(1,1,0) \oplus f(1,1,1)] \\
& \oplus \bar{x}_{3}[f(0,0,0) \oplus f(1,0,0)] \\
& \oplus \bar{x}_{3} x_{1}[f(0,0,0) \oplus f(0,0,1) \oplus f(1,0,0) \oplus f(1,0,1)] \\
& \oplus \bar{x}_{3} x_{2}[f(0,0,0) \oplus f(0,1,0) \oplus f(1,0,0) \oplus f(1,1,0)] \\
& \oplus \bar{x}_{3} x_{2} x_{1}[f(0,0,0) \oplus f(0,0,1) \oplus f(0,1,0) \oplus f(0,1,1) \\
& \oplus f(1,0,0) \oplus f(1,0,1) \oplus f(1,1,0) \oplus f(1,1,1)]
\end{aligned}
$$

### 2.5 Classes of Exclusive-OR Sum-of-Products Expansions

The FPRM expansions of switching functions, as defined in Equation (2.5), constitute only a small subclass of the total number of exclusive-OR (or ring) sum-of-products forms which uniquely describe any switching function. ESOP forms may be divided into several categories, where each category contains expansions which display similar characteristics. These range from the well-defined, consistent and canonical FPRM forms to the inconsistent generalised Reed-Muller (GRM) expansions (defined in section 2.5.3). Figure 2.2 illustrates the classes of ESOP expansions which may represent any switching function and their relationships. This diagram was presented by Sasao [7] and the definitions adopted in this thesis are those proposed by Sasao.

It is, perhaps, useful to relate the notation introduced by Sasao, and employed in this thesis, to another popular form of notation [27].

Alternative name [27]

PPRM (Positive Polarity RM) expansion (Equation (2.4))
FPRM (Fixed Polarity RM) expansions GRM (generalised RM) expansions (Equation (2.5))
KRO (KROnecker) expansions KRM (Kronecker RM) expansions PSDRM (PSeuDo RM) expansions $\}$ PKRM (Pseudo Kronecker RM) PSDKRO (PSeuDo KROnecker) expansions

RM expansion $\}$ PKRM $\begin{gathered}\text { (Pseudo Kronecker RM) } \\ \text { expansions }\end{gathered}$

The categories of ESOP forms which have been introduced are now considered in more detail.

### 2.5.1 Kronecker Expansions

The Kronecker expansions may be termed mixed polarity RM expansions as each expansion variable may appear in both true and complemented forms throughout an expression. A KRO expansion is constructed from an initial switching function by expanding the function (subfunction) about each variable using one of the expansions given in Equations (2.14), (2.15) and (2.16). There are $3^{n}$ possible combinations of these 3 equations, hence any $n$ variable switching function may be represented by a total of $3^{n}$ KRO expansions, each of which is a canonical form. The RM expansion and FPRM expansions which may be constructed by employing only expansion (2.15) and combinations of expansions (2.15) and (2.16) respectively, are also KRO
expansions. This is illustrated in Figure 2.2. As a general rule, a KRO expansion may be identified by observing that an expansion variable which appears in both true and complemented forms throughout the expression must be present in each and every product term. This constraint is relaxed for expansion variables which consistently appear in either true or complemented form, but not both forms.

Key
PPRM - Positive Polarity Reed-Muller expansion
FPRM - Fixed Polarity Reed-Muller expansions
KRO - KROnecker expansions
PSDRM - PSeuDo Reed-Muller expansions
PSDKRO - PSeuDo KROnecker expansions
GRM - Generalised Reed-Muller expansions
ESOP - Exclusive-OR Sum-Of-Products expansions

Figure 2.2: Classes of Exclusive-OR sum-of-products expansions and their relationships.

Each KRO expansion $f\left(x_{n}, x_{n-1}, \ldots, x_{1}\right)$ may be identified by means of a polarity number $m, 0 \leq m \leq 3^{n}-1$. Hence, the polarity $m$ KRO expansion is denoted $f_{m}\left(x_{n}, x_{n-1}, \ldots, x_{1}\right)$. This number, $m$, indicates the state of each expansion
variable throughout the expression, that is, which variables are present in true form, complemented form or both true and complemented forms. The polarity number $m$ is the decimal equivalent of the ternary $n$-tuple $<m_{n} m_{n-1}$ $\ldots m_{1}>$, where $m_{j}$ is replaced by 0 if $x_{j}$ is present throughout the KRO expansion, and replaced by 1 if $\bar{x}_{j}$ is consistently present in the expansion. If the variable is present in both true and complemented forms, i.e. $x_{j}$ and $\bar{x}_{j}$, then $m_{j}$ is replaced by 2 . The FPRM expansions will correspond to all polarity numbers whose ternary forms comprise only 0 's and 1 's.

The following expressions illustrate the use of polarity numbers to identify KRO expansions.
e.g. for $n=3$,

$$
m=0=\langle 000\rangle \quad \begin{aligned}
& m_{3}=0 \rightarrow x_{3} \\
& m_{2}=0 \Rightarrow x_{2} \\
& m_{1}=0 \Rightarrow x_{1}
\end{aligned}
$$

Polarity 0 KRO expansion. (This is also the polarity 0 FPRM expansion (Equation (2.9)), and is termed the positive polarity RM expansion.)

$$
\begin{aligned}
& f_{0}\left(x_{3}, x_{2}, x_{1}\right)= f(0,0,0) \oplus x_{1}[f(0,0,0) \oplus f(0,0,1)] \oplus x_{2}[f(0,0,0) \oplus f(0,1,0)] \\
& \oplus x_{2} x_{1}[f(0,0,0) \oplus f(0,0,1) \oplus f(0,1,0) \oplus f(0,1,1)] \oplus x_{3}[f(0,0,0) \oplus f(1,0,0)] \\
& \oplus x_{3} x_{1}[f(0,0,0) \oplus f(0,0,1) \oplus f(1,0,0) \oplus f(1,0,1)] \\
& \oplus x_{3} x_{2}[f(0,0,0) \oplus f(0,1,0) \oplus f(1,0,0) \oplus f(1,1,0)] \\
& \oplus x_{3} x_{2} x_{1}[f(0,0,0) \oplus f(0,0,1) \oplus f(0,1,0) \oplus f(0,1,1) \\
&\oplus f(1,0,0) \oplus f(1,0,1) \oplus f(1,1,0) \oplus f(1,1,1)] \\
& m=1=<001>\quad m_{3}=0 \rightarrow x_{3} \\
& m_{2}=0 \rightarrow x_{2} \\
& m_{1}=1 \rightarrow \bar{x}_{1}
\end{aligned}
$$

Polarity 1 KRO expansion (and the polarity 1 FPRM expansion).

$$
\begin{aligned}
f_{1}\left(x_{3}, x_{2}, x_{1}\right)= & f(0,0,1) \oplus \bar{x}_{1}[f(0,0,0) \oplus f(0,0,1)] \oplus x_{2}[f(0,0,1) \oplus f(0,1,1)] \\
& \oplus x_{2} \bar{x}_{1}[f(0,0,0) \oplus f(0,0,1) \oplus f(0,1,0) \oplus f(0,1,1)] \oplus x_{3}[f(0,0,1) \oplus f(1,0,1)] \\
& \oplus x_{3} \bar{x}_{1}[f(0,0,0) \oplus f(0,0,1) \oplus f(1,0,0) \oplus f(1,0,1)] \\
& \oplus x_{3} x_{2}[f(0,0,1) \oplus f(0,1,1) \oplus f(1,0,1) \oplus f(1,1,1)] \\
& \oplus x_{3} x_{2} \bar{x}_{1}[f(0,0,0) \oplus f(0,0,1) \oplus f(0,1,0) \oplus f(0,1,1) \\
& \oplus f(1,0,0) \oplus f(1,0,1) \oplus f(1,1,0) \oplus f(1,1,1)]
\end{aligned}
$$

$$
m=2=<002>\quad \begin{aligned}
& m_{3}=0 \rightarrow x_{3} \\
& m_{2}=0 \rightarrow x_{2} \\
& m_{1}=2 \rightarrow x_{1}, \bar{x}_{1}
\end{aligned}
$$

Polarity 2 KRO expansion.

$$
\begin{aligned}
f_{2}\left(x_{3}, x_{2}, x_{1}\right)= & \bar{x}_{1} f(0,0,0) \oplus x_{1} f(0,0,1) \oplus x_{2} \bar{x}_{1}[f(0,0,0) \oplus f(0,1,0)] \\
& \oplus x_{2} x_{1}[f(0,0,1) \oplus f(0,1,1)] \oplus x_{3} \bar{x}_{1}[f(0,0,0) \oplus f(1,0,0)] \\
& \oplus x_{3} x_{1}[f(0,0,1) \oplus f(1,0,1)] \\
& \oplus x_{3} x_{2} \bar{x}_{1}[f(0,0,0) \oplus f(0,1,0) \oplus f(1,0,0) \oplus f(1,1,0)] \\
& \oplus x_{3} x_{2} x_{1}[f(0,0,1) \oplus f(0,1,1) \oplus f(1,0,1) \oplus f(1,1,1)] \\
m=4=<011>\quad & m_{3}=0 \rightarrow x_{3} \\
& m_{2}=1 \rightarrow \bar{x}_{2} \\
& m_{1}=1 \rightarrow \bar{x}_{1}
\end{aligned}
$$

Polarity 4 KRO expansion (and the polarity 3 FPRM expansion (Equation (2.10))).

$$
\begin{aligned}
f_{1}\left(x_{3}, x_{2}, x_{1}\right)= & f(0,1,1) \oplus \bar{x}_{1}[f(0,1,0) \oplus f(0,1,1)] \oplus \bar{x}_{2}[f(0,0,1) \oplus f(0,1,1)] \\
& \oplus \bar{x}_{2} \bar{x}_{1}[f(0,0,0) \oplus f(0,0,1) \oplus f(0,1,0) \oplus f(0,1,1)] \oplus x_{3}[f(0,1,1) \oplus f(1,1,1)] \\
& \oplus x_{3} \bar{x}_{1}[f(0,1,0) \oplus f(0,1,1) \oplus f(1,1,0) \oplus f(1,1,1)] \\
& \oplus x_{3} \bar{x}_{2}[f(0,0,1) \oplus f(0,1,1) \oplus f(1,0,1) \oplus f(1,1,1)] \\
& \oplus x_{3} \bar{x}_{2} \bar{x}_{1}[f(0,0,0) \oplus f(0,0,1) \oplus f(0,1,0) \oplus f(0,1,1) \\
& \oplus f(1,0,0) \oplus f(1,0,1) \oplus f(1,1,0) \oplus f(1,1,1)]
\end{aligned}
$$

$$
m=5=<012>\quad \begin{aligned}
& m_{3}=0 \rightarrow x_{3} \\
& m_{2}=1 \rightarrow \bar{x}_{2} \\
& m_{1}=2 \rightarrow x_{1}, \bar{x}_{1}
\end{aligned}
$$

Polarity 5 KRO expansion.

$$
\begin{aligned}
f_{3}\left(x_{3}, x_{2}, x_{1}\right)= & \bar{x}_{1} f(0,1,0) \oplus x_{1} f(0,1,1) \oplus \bar{x}_{2} \bar{x}_{1}[f(0,0,0) \oplus f(0,1,0)] \\
& \oplus \bar{x}_{2} x_{1}[f(0,0,1) \oplus f(0,1,1)] \oplus x_{3} \bar{x}_{1}[f(0,1,0) \oplus f(1,1,0)] \\
& \oplus x_{3} x_{1}[f(0,1,1) \oplus f(1,1,1)] \\
& \oplus \bar{x}_{3} \bar{x}_{2} \bar{x}_{1}[f(0,0,0) \oplus f(0,1,0) \oplus f(1,0,0) \oplus f(1,1,0)] \\
& \oplus x_{3} \bar{x}_{2} x_{1}[f(0,0,1) \oplus f(0,1,1) \oplus f(1,0,1) \oplus f(1,1,1)] \\
m=13=<111\rangle \quad & m_{3}=1 \rightarrow \bar{x}_{3} \\
& m_{2}=1 \rightarrow \bar{x}_{2} \\
& m_{1}=1 \rightarrow \bar{x}_{1}
\end{aligned}
$$

Polarity 13 KRO expansion. (As this is also the polarity 7 FPRM expansion (Equation (2.11)) it may be termed the negative polarity RM expansion.)

$$
\begin{aligned}
f_{13}\left(x_{3}, x_{2}, x_{1}\right)= & f(1,1,1) \oplus \bar{x}_{1}[f(1,1,0) \oplus f(1,1,1)] \\
& \oplus \bar{x}_{2}[f(1,0,1) \oplus f(1,1,1)] \\
& \oplus \bar{x}_{2} \bar{x}_{1}[f(1,0,0) \oplus f(1,0,1) \oplus f(1,1,0) \oplus f(1,1,1)] \\
& \oplus \bar{x}_{3}[f(0,1,1) \oplus f(1,1,1)] \\
& \oplus \bar{x}_{3} \bar{x}_{1}[f(0,1,0) \oplus f(0,1,1) \oplus f(1,1,0) \oplus f(1,1,1)] \\
& \oplus \bar{x}_{3} \bar{x}_{2}[f(0,0,1) \oplus f(0,1,1) \oplus f(1,0,1) \oplus f(1,1,1)] \\
& \oplus \bar{x}_{3} \bar{x}_{2} \bar{x}_{1}[f(0,0,0) \oplus f(0,0,1) \oplus f(0,1,0) \oplus f(0,1,1) \\
& \oplus f(1,0,0) \oplus f(1,0,1) \oplus f(1,1,0) \oplus f(1,1,1)]
\end{aligned}
$$

$$
m=26=<222>\quad m_{3}=2 \quad \rightarrow \quad x_{3}, \bar{x}_{3}
$$

$$
m_{2}=2 \rightarrow x_{2}, \bar{x}_{2}
$$

$$
m_{1}=2 \Rightarrow x_{1}, \bar{x}_{1}
$$

Polarity 26 KRO expansion.

$$
\begin{aligned}
f_{20}\left(x_{3}, x_{2}, x_{1}\right)= & \bar{x}_{3} \bar{x}_{2} \bar{x}_{1} f(0,0,0) \oplus \bar{x}_{3} \bar{x}_{2} x_{1} f(0,0,1) \oplus \bar{x}_{3} x_{2} \bar{x}_{1} f(0,1,0) \\
& \oplus \bar{x}_{3} x_{2} x_{1} f(0,1,1) \oplus x_{3} \bar{x}_{2} \bar{x}_{1} f(1,0,0) \oplus x_{3} \bar{x}_{2} x_{1} f(1,0,1) \\
& \oplus x_{3} x_{2} \bar{x}_{1} f(1,1,0) \oplus x_{3} x_{2} x_{1} f(1,1,1)
\end{aligned}
$$

Note that the polarity $26\left(3^{n}-1\right) \mathrm{KRO}$ expansion, with all variables present in both true and complemented forms, is equivalent to the Boolean SOP expansion (Equation (2.2)).

### 2.5.2 Pseudo Reed-Muller Expansions and Pseudo Kronecker Expansions

 Pseudo Reed-Muller expansions and Pseudo Kronecker expansions exhibit similarities in their basic structure. Both types of expressions may be constructed using the same technique and will therefore be described simultaneously.The Shannon expansion theorem (Equation (2.12)) may be modified to allow a switching function to be represented using three different expansions. This was demonstrated in section 2.4 and the expressions are now repeated.

$$
\begin{equation*}
f\left(x_{n}, x_{n-1}, \ldots, x_{1}\right)=\bar{x}_{j} f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{2} x_{1}\right) \oplus x_{j} f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{2}, x_{1}\right) \tag{2.17}
\end{equation*}
$$

$$
\begin{align*}
f\left(x_{n}, x_{n-1}, \ldots, x_{1}\right)= & f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{2}, x_{1}\right) \\
& \oplus x_{j}\left[f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{2} x_{1}\right) \oplus f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{2}, x_{1}\right)\right] \tag{2.18}
\end{align*}
$$

$$
\begin{align*}
f\left(x_{n}, x_{n-1}, \ldots, x_{1}\right)= & f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{2}, x_{1}\right) \\
& \bullet \bar{x}_{j}\left[f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{2} x_{1}\right) \oplus f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{2}, x_{1}\right)\right] \tag{2.19}
\end{align*}
$$

The original function $f\left(x_{n}, x_{n-1}, \ldots, x_{1}\right)$ is now composed of two subfunctions of ( $n-1$ ) variables. Each subfunction may be considered to be a coefficient of the expansion, and is either independent of the function variable $x_{j}$ or is associated with literal $x_{j}$ or $\bar{x}_{j}$. There are a total of 3 possible subfunctions and of these any 2 are used in the representation of the original switching function. The subfunctions are

$$
\begin{equation*}
f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{2} x_{1}\right) \tag{2.20}
\end{equation*}
$$

$$
\begin{equation*}
f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{2}, x_{1}\right) \tag{2.21}
\end{equation*}
$$

$$
\begin{equation*}
f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{2} x_{1}\right) \oplus f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{2}, x_{1}\right) \tag{2.22}
\end{equation*}
$$

Each subfunction may be expanded about function variable $x_{k}(k=1,2, \ldots, n$ $k \neq j$ ). The original subfunction is split into 2 new subfunctions of ( $n-2$ ) variables, which exhibit structures similar to the subfunctions denoted in Equations (2.20), (2.21) and (2.22).

The PSDRM expansion is constructed by applying either expansion (2.18) or (2.19) to the switching function $f\left(x_{n}, x_{n-1}, \ldots, x_{1}\right)$. Either of these expansions are then applied to each subfunction until the function is expanded about all $n$ variables. It is possible to construct $2^{2^{n}-1}$ canonic PSDRM expansions. PSDKRO expansions are constructed in a similar manner, allowing expansions
(2.17), (2.18) and (2.19) to be applied to the switching function and to all subsequent subfunctions. This results in a total of $3^{2^{n}-1}$ possible canonic PSDKRO expansions.

Note that in constructing PSDRM and PSDKRO expansions it is not necessary to apply the same expansion (i.e. (2.17), (2.18) or (2.19)) to each subfunction. If, however, the same expansion is applied to all subfunctions of any function variable and this is adhered to for each function variable then the expression constructed is a KRO expansion. If a further constraint is imposed and only expansions (2.18) and (2.19) may be applied then a FPRM expansion will be formed. Hence FPRM expansions are a subclass of PSDRME expansions whilst KRO forms are a subclass of PSDKRO expansions. Additionally, PSDKRO forms include all PSDRME expansions. These relationships are illustrated in Figure 2.2. Henceforth, any reference to PSDKRO expansions will include all PSDRM expansions.

A PSDKRO expansion $f\left(x_{n}, x_{n-1}, \ldots, x_{1}\right)$ may be identified by a polarity number $q, 0 \leq q \leq 3^{2^{n}-1}-1$. The polarity $q$ PSDKRO expansion may be denoted $f_{q}\left(x_{n}, x_{n-1}, \ldots, x_{1}\right)$. The number $q$ is the decimal equivalent of the ternary ( $2^{n}-1$ )-bit number and each $q_{k}$ indicates the type of expansion, i.e. (2.17), (2.18) or (2.19), which should be applied to each subfunction of $j$ variables. The ternary digit $q_{2^{n}-1}$ indicates which expansion should be applied to the subfunction of $n$ variables, i.e. the original function. Ternary digits $q_{2} n_{-2}$ and $q_{2^{n}-3}$ dictate which expansion should be applied to the two subfunctions of ( $n-1$ ) variables. If expansion (2.17) is applied to the original function then $g_{2^{n}-3}$ indicates which expansion should be applied to the subfunction associated with literal $\bar{x}_{n}$, whilst $\boldsymbol{q}_{2^{n}-2}$ indicates which expansion should be applied to the subfunction associated with literal $x_{n}$. Alternatively, if expansion (2.18) ((2.19)) is applied to the original function then $g_{2^{n}-3}$ indicates which expansion should be applied to the subfunction which is independent of literal $x_{n}\left(\bar{x}_{n}\right)$, whilst $g_{2^{n}-2}$ indicates which expansion should be applied to the subfunction associated with literal $x_{n}$ $\left(\bar{x}_{n}\right)$. This rule may be applied for each function variable and the associated
 which expansion should be applied to the subfunction of variable $j(j=$ $1,2, \ldots, n$ ).

The use of the polarity number, $q$, is illustrated in the following examples, e.g. for $n=3$,

$$
q=270=<0101000>\quad \begin{aligned}
& q_{7}=0 \rightarrow x_{3} \\
& q_{6}=1 \rightarrow \bar{x}_{2} \\
& q_{5}=0 \Rightarrow x_{2} \\
& q_{4}=1 \Rightarrow \bar{x}_{1} \\
& q_{3}=0 \Rightarrow x_{1} \\
& q_{2}=0 \Rightarrow x_{1} \\
& q_{1}=0 \Rightarrow x_{1}
\end{aligned}
$$

Polarity 270 PSDKRO expansion.
$q_{7}=0$, therefore apply expansion (2.18) to the original function.

$$
f\left(x_{3}, x_{2}, x_{1}\right)=f\left(0, x_{2}, x_{1}\right) \oplus x_{3}\left[f\left(0, x_{2}, x_{1}\right) \oplus f\left(1, x_{2}, x_{1}\right)\right]
$$

$q_{6}=1$, therefore apply expansion (2.19) to the subfunction associated with literal $x_{3}$.
$q_{5}=0$, therefore apply expansion (2.18) to the subfunction independent of literals $X_{3}, \bar{x}_{3}$.

$$
\begin{aligned}
f\left(x_{3}, x_{2}, x_{1}\right)= & f\left(0,0, x_{1}\right) \oplus x_{2}\left[f\left(0,0, x_{1}\right) \oplus f\left(0,1, x_{1}\right)\right] \\
& \oplus x_{3}\left[f\left(0,1, x_{1}\right) \oplus \bar{x}_{2}\left[f\left(0,0, x_{1}\right) \oplus f\left(0,1, x_{1}\right)\right]\right. \\
& \left.\oplus f\left(1,1, x_{1}\right) \oplus \bar{x}_{2}\left[f\left(1,0, x_{1}\right) \oplus f\left(1,1, x_{1}\right)\right]\right]
\end{aligned}
$$

Rearrange

$$
\begin{aligned}
f\left(x_{3}, x_{2}, x_{1}\right)= & f\left(0,0, x_{1}\right) \oplus x_{2}\left[f\left(0,0, x_{1}\right) \oplus f\left(0,1, x_{1}\right)\right] \\
& \oplus x_{3}\left[\left[f\left(0,1, x_{1}\right) \oplus f\left(1,1, x_{1}\right)\right]\right. \\
& \left.\oplus \bar{x}_{2}\left[f\left(0,0, x_{1}\right) \oplus f\left(0,1, x_{1}\right) \oplus f\left(1,0, x_{1}\right) \oplus f\left(1,1, x_{1}\right)\right]\right]
\end{aligned}
$$

$q_{4}=1$, therefore apply expansion (2.19) to the subfunction associated with literals $x_{3} \bar{x}_{2}$.
$q_{3}=0$, therefore apply expansion (2.18) to the subfunction associated with literal $x_{3}$.
$q_{2}=0$, therefore apply expansion (2.18) to the subfunction associated with literal $x_{2}$.
$q_{1}=0$, therefore apply expansion (2.18) to the subfunction independent of any literals.
The polarity 270 PSDKRO expansion may be formed by expanding each subfunction about variable $x_{1}$ using the appropriate expressions for each
subfunction.
Thus,

$$
\begin{aligned}
f_{270}\left(x_{3}, x_{2}, x_{1}\right)= & f(0,0,0) \oplus x_{1}[f(0,0,0) \oplus f(0,0,1)] \\
& \oplus x_{2}\left[f(0,0,0) \oplus x_{1}[f(0,0,0) \oplus f(0,0,1)] \oplus f(0,1,0) \oplus x_{1}[f(0,1,0) \oplus f(0,1,1)]\right] \\
& \oplus x_{3}\left[f(0,1,0) \oplus x_{1}[f(0,1,0) \oplus f(0,1,1)] \oplus f(1,1,0) \oplus x_{1}[f(1,1,0) \oplus f(1,1,1)]\right. \\
& \oplus \bar{x}_{2}\left[f(0,0,1) \oplus \bar{x}_{1}[f(0,0,0) \oplus f(0,0,1)] \oplus f(0,1,1) \oplus \bar{x}_{1}[f(0,1,0) \oplus f(0,1,1)]\right. \\
& \left.\left.\oplus f(1,0,1) \oplus \bar{x}_{1}[f(1,0,0) \oplus f(1,0,1)] \oplus f(1,1,1) \oplus \bar{x}_{1}[f(1,1,0) \oplus f(1,1,1)]\right]\right]
\end{aligned}
$$

Rearrange

$$
\begin{aligned}
f_{2 \sim}\left(x_{3}, x_{2}, x_{1}\right)= & f(0,0,0) \oplus x_{1}[f(0,0,0) \oplus f(0,0,1)] \oplus x_{2}[f(0,0,0) \oplus f(0,1,0)] \\
& \oplus x_{2} x_{1}[f(0,0,0) \oplus f(0,0,1) \oplus f(0,1,0) \oplus f(0,1,1)] \oplus x_{3}[f(0,1,0) \oplus f(1,1,0)] \\
& \oplus x_{3} x_{1}[f(0,1,0) \oplus f(0,1,1) \oplus f(1,1,0) \oplus f(1,1,1)] \\
& \oplus x_{3} \bar{x}_{2}[f(0,0,1) \oplus f(0,1,1) \oplus f(1,0,1) \oplus f(1,1,1)] \\
& \oplus x_{3} \bar{x}_{2} \bar{x}_{1}[f(0,0,0) \oplus f(0,0,1) \oplus f(0,1,0) \oplus f(0,1,1) \\
& \oplus f(1,0,0) \oplus f(1,0,1) \oplus f(1,1,0) \oplus f(1,1,1)]
\end{aligned}
$$

This PSDKRO expansion is also a PSDRM expansion as only expansions (2.18) and (2.19) are applied.

$$
\begin{aligned}
& q=1328=<1211012>\quad q_{7}=1 \quad \Rightarrow \bar{x}_{3} \\
& q_{6}=2 \rightarrow x_{2}, \bar{x}_{2} \\
& q_{5}=1 \rightarrow \bar{x}_{2} \\
& q_{4}=1 \rightarrow \bar{x}_{1} \\
& q_{3}=0 \rightarrow x_{1} \\
& q_{2}=1 \rightarrow \bar{x}_{1} \\
& q_{1}=2 \rightarrow x_{1}, \bar{x}_{1}
\end{aligned}
$$

Polarity 1328 PSDKRO expansion. (This expansion is also derived in [27]) $q_{7}=1$, therefore apply expansion (2.19) to the original function.

$$
f\left(x_{3}, x_{2}, x_{1}\right)=f\left(1, x_{2}, x_{1}\right) \oplus \bar{x}_{3}\left[f\left(0, x_{2}, x_{1}\right) \oplus f\left(1, x_{2}, x_{1}\right)\right]
$$

$q_{6}=2$, therefore apply expansion (2.17) to the subfunction associated with literal $\bar{x}_{3}$.
$q_{5}=1$, therefore apply expansion (2.19) to the subfunction independent of
literals $x_{3}, \bar{x}_{3}$.

$$
\begin{aligned}
f\left(x_{3}, x_{2}, x_{1}\right)= & f\left(1,1, x_{1}\right) \oplus \bar{x}_{2}\left[f\left(1,0, x_{1}\right) \oplus f\left(1,1, x_{1}\right)\right] \\
& \oplus \bar{x}_{3}\left[\bar{x}_{2} f\left(0,0, x_{1}\right) \oplus x_{2} f\left(0,1, x_{1}\right) \oplus \bar{x}_{2} f\left(1,0, x_{1}\right) \oplus x_{2} f\left(1,1, x_{1}\right)\right]
\end{aligned}
$$

Rearrange

$$
\begin{aligned}
f\left(x_{3}, x_{2}, x_{1}\right)= & f\left(1,1, x_{1}\right) \oplus \bar{x}_{2}\left[f\left(1,0, x_{1}\right) \oplus f\left(1,1, x_{1}\right)\right] \\
& \oplus \bar{x}_{3}\left[\bar{x}_{2}\left[f\left(0,0, x_{1}\right) \oplus f\left(1,0, x_{1}\right)\right] \oplus x_{2}\left[f\left(0,1, x_{1}\right) \oplus f\left(1,1, x_{1}\right)\right]\right]
\end{aligned}
$$

$q_{4}=1$, therefore apply expansion (2.19) to the subfunction associated with literals $\bar{x}_{3} x_{2}$.
$q_{3}=0$, therefore apply expansion (2.18) to the subfunction associated with literal $\bar{x}_{3} \bar{x}_{2}$.
$q_{2}=1$, therefore apply expansion (2.19) to the subfunction associated with literal $\bar{x}_{2}$.
$q_{1}=2$, therefore apply expansion (2.17) to the subfunction independent of any literals.
The polarity 1328 PSDKRO expansion may be formed by expanding each subfunction about variable $x_{1}$ using the appropriate expressions for each subfunction.

Thus,

$$
\begin{aligned}
f_{1328}\left(x_{3}, x_{2}, x_{1}\right)= & \bar{x}_{1} f(1,1,0) \oplus x_{1} f(1,1,1) \\
& \oplus \bar{x}_{2}\left[f(1,0,1) \oplus \bar{x}_{1}[f(1,0,0) \oplus f(1,0,1)]\right. \\
& \left.\oplus f(1,1,1) \oplus \bar{x}_{1}[f(1,1,0) \oplus f(1,1,1)]\right] \\
& \oplus \bar{x}_{3}\left[\overline { x } _ { 2 } \left[f(0,0,0) \oplus x_{1}[f(0,0,0) \oplus f(0,0,1)]\right.\right. \\
& \left.\bullet f(1,0,0) \oplus x_{1}[f(1,0,0) \oplus f(1,0,1)]\right] \\
& \oplus x_{2}\left[f(0,1,1) \oplus \bar{x}_{1}[f(0,1,0) \oplus f(0,1,1)]\right. \\
& \left.\left.\oplus f(1,1,1) \oplus \bar{x}_{1}[f(1,1,0) \oplus f(1,1,1)]\right]\right]
\end{aligned}
$$

Rearrange

$$
\begin{aligned}
f_{1328}\left(x_{3}, x_{2}, x_{1}\right)= & \bar{x}_{1} f(1,1,0) \oplus x_{1} f(1,1,1) \oplus \bar{x}_{2}[f(1,0,1) \oplus f(1,1,1)] \\
& \oplus \bar{x}_{2} \bar{x}_{1}[f(1,0,0) \oplus f(1,0,1) \oplus f(1,1,0) \oplus f(1,1,1)] \\
& \oplus \bar{x}_{3} \bar{x}_{2}[f(0,0,0) \oplus f(1,0,0)] \\
& \oplus \bar{x}_{3} \bar{x}_{2} x_{1}[f(0,0,0) \oplus f(0,0,1) \oplus f(1,0,0) \oplus f(1,0,1)] \\
& \oplus \bar{x}_{3} x_{2}[f(0,1,1) \oplus f(1,1,1)] \\
& \oplus \bar{x}_{3} \bar{x}_{2} \bar{x}_{1}[f(0,1,0) \oplus f(0,1,1) \oplus f(1,1,0) \oplus f(1,1,1)]
\end{aligned}
$$

### 2.5.3 Generalised Reed-Muller Expansions

The FPRM expansions defined in section 2.3 exhibit the basic property that each expansion variable appears in either true or complemented form throughout the expression. These expressions may therefore be termed consistent canonical forms [28]. Another class of exclusive-OR sum-ofproducts expansions may be formed by considering the RM expansion (PPRM expansion),

$$
f\left(x_{n}, x_{n-1}, \ldots, x_{1}\right)=a_{0} \oplus a_{1} x_{1} \oplus a_{2} x_{2} \oplus a_{3} x_{2} x_{1} \oplus \ldots \ldots \oplus a_{2 n-1} x_{n} x_{n-1} \ldots x_{2} x_{1}
$$

This expression may be modified by replacing any combination of literals by their complemented forms (employing the substitution $x_{j}=\bar{x}_{j} \oplus 1$ ). This allows variables to appear in both true and complemented form throughout some expansions and these are termed inconsistent canonical forms [28]. The total number of possible combinations of true and complemented literals is $2^{n 2^{n-1}}$. It is, therefore, possible to derive $2^{n 2^{n-1}}$ expressions, termed generalised Reed-Muller (GRM) expansions. The GRM expansions are comprised of the $2^{n}$ consistent canonical forms, which are the FPRM expansions, and $2^{n 2^{n-1}}-2^{n}$ inconsistent canonical forms. This relationship is illustrated in Figure 2.2.

### 2.5.4 Exclusive-OR Sum-of-Products Expansions

Exclusive-OR sum-of-products expansions encompass any switching function which employs modulo-2 operators. This obviously includes all canonical expansions defined in the categories of the preceding sections and any canonical expansions which do not fall into any of these categories. Additionally, modulo-2 sum-of-products expansions which are not canonical
forms may be classified under the general title of exclusive-OR sum-ofproducts expansions. It is possible to derive a maximum of $3^{\text {tn }}$ ESOP forms of $n$ expansion variables and $t$ product terms [7]. This category is very general and many of these ESOP forms have little regular structure and are loosely defined.

### 2.6 Binary Decision Diagrams

Binary Decision Trees and Binary Decision Diagrams [29, 30, 31] are graphical representations of switching function and are an alternative to Karnaugh maps and truth tables. The BDT of the $n$ variable Boolean function

$$
\begin{align*}
f\left(x_{n}, x_{n-1}, \ldots, x_{1}\right)= & d_{0} \bar{x}_{n} \bar{x}_{n-1} \ldots \bar{x}_{2} \bar{x}_{1}+d_{1} \bar{x}_{n} \bar{x}_{n-1} \ldots \bar{x}_{2} x_{1}+d_{2} \bar{x}_{n} \bar{x}_{n-1} \ldots x_{2} \bar{x}_{1}+\ldots \\
& \ldots+d_{2-1} x_{n} x_{n-1} \ldots x_{2} x_{1} \tag{2.23}
\end{align*}
$$

is illustrated in Figure 2.3. The Boolean SOP expansion of Equation (2.23) is a canonical form where $d_{1} \in\{0,1\}$ are the operational domain coefficients ( $i=0,1, \ldots, 2^{n}-1$ ) and $x_{j}$ and $\bar{x}_{j}$ are literals of the expansion, in true and complemented forms respectively ( $j=1,2, \ldots, n$ ).

A BDT (Figure 2.3) is comprised of nodes connected to one another by branches. Two types of nodes are present in the structure, namely terminal nodes and non-terminal nodes. A terminal node (box) may assume either the value 0 or the value 1 , whilst each non-terminal node (circle) is associated with a function variable. Every non-terminal node has one input branch and two output branches. The left output branch, denoted 0 , indicates the presence of the node variable in complemented form. The right output branch is denoted 1 , indicating that the node variable is present in true form.

The function of a non-terminal node is illustrated in Figure 2.4. Hence, the node variable $x_{j}(j=1,2, \ldots, n)$ may be considered to be the splitting variable in the Shannon expansion theorem. The functions $f\left(x_{n}, x_{n-1}\right.$, $\left.\ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{2}, x_{1}\right)$ and $f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{2}, x_{1}\right)$ are independent of variable $x_{j}$ and are subfunctions of the original Boolean function. A subtree is defined as a part of a BDT and is, therefore, also a BDT. Any subtree represents a subfunction of the original function depicted by the full BDT.

A subtree will be comprised of fewer nodes than the BDT of the original function and may be rooted at an output branch of any non-terminal node of the BDT.


Figure 2.3: Ordered Binary Decision Tree of a $n$ variable Boolean function


Figure 2.4: Non-terminal node representing function variable $x_{j}$

Tracing a path from the root of the BDT (Figure 2.3) to a terminal node with value 1 realises a minterm of the Boolean function. If this is repeated for all terminal nodes with value 1 then the canonical Boolean SOP expression is formed.

Branches of a BDT may only connect nodes on level $k$ to nodes on level $m$, where $k<m$ and $k, m \in\{1,2, \ldots, n\}$. The BDTs discussed in this thesis are ordered structures. That is, if a path is traced from the root to the terminal nodes each function variable will be encountered only once. The order in which the function variables are encountered is identical for each path traced from the root of the BDT to a terminal node. Relating this to Figure 2.3, each node at level $j$ will represent the same function variable $x_{k}(j, k \in\{1,2, \ldots, n\})$. Henceforth, only Ordered Binary Decision Trees (OBDTs) will be considered.

The OBDT of any $n$ variable function is comprised of a total of $2^{n+1}-1$ nodes of which $2^{n}-1$ are non-terminal nodes. The remaining number of terminal nodes ( $2^{n}$ ) is equal to the number of rows in the truth table representation of a $n$ variable function. It is, however, possible to reduce the number of nodes in a BDT by deleting redundant nodes and merging identical subtrees.

Equivalent nodes [32]
Two terminal nodes of an OBDT are equivalent if they each have the same Boolean value.

Two non-terminal nodes of an OBDT are equivalent if both nodes represent the same function variable, the subtrees rooted at the left output branch of the nodes are identical and subtrees rooted at the right output branch are identical.

The definition of equivalent nodes leads to the formulation of two rules which may be employed to reduce the number of nodes in an OBDT.

Reduction rules [32]

1) If the subtree (terminal node) rooted at the left output branch of a node is equivalent to the subtree (terminal node) rooted at the right output branch then redirect the input of the node to the input branch of the left subtree (terminal node). Delete the node and the right subtree (terminal node).
2) If two nodes, $a$ and $b$, of a OBDT are equivalent, redirect the input branch of node $b$ to the input of node $a$. Node $b$ and its subtree can then be deleted.
where the nodes of an OBDD may have more than one input branch connected to any node. If the reduction rules which have been detailed are repeatedly applied to an OBDT until the number of nodes in the structure can no longer be reduced then a Reduced Ordered Binary Decision Diagram (ROBDD) is formed. The ROBDD comprises of a minimum number of nodes for a given variable ordering, and is a unique representation of the Boolean function. A ROBDD can, therefore, be defined as a canonical representation of a Boolean function [31].

A ROBDD represents the essential implicants of a Boolean function. The implicants are essential as every minterm (product term) of the initial Boolean function is covered by one implicant represented by the ROBDD. It cannot, however, be guaranteed that the implicants are prime implicants as the ROBDD representation does not allow minterms (product terms) to be covered by more than one implicant [33]. Hence, the implicants represented by a ROBDD are disjoint.

The use of the reduction rules is illustrated in Figure 2.5, where the Boolean function

$$
f\left(x_{3}, x_{2}, x_{1}\right)=\bar{x}_{3} \bar{x}_{2} \bar{x}_{1}+x_{3} \bar{x}_{2} \bar{x}_{1}+x_{3} \bar{x}_{2} x_{1}+x_{3} x_{2} \bar{x}_{1}+x_{3} x_{2} x_{1}
$$

is first represented by an OBDT (Figure 2.5(a)). The order of the function variables in Figure 2.5(a)-Figure 2.5(c) is $\langle 3,2,1\rangle$, starting with the lowest level of non-terminal nodes and ending at the root of the structure. The OBDD of Figure 2.5(b) illustrates equivalent nodes which may be merged and identifies redundant nodes. The ROBDD representing the Boolean function is shown in Figure 2.5(c). This ROBDD is comprised of 3 nonterminal nodes and 2 terminal nodes. Thus, the total of nodes has been reduced from 15 in the OBDT to 5 in the ROBDD. The number of paths which terminate in a node with value one has been reduced from 5 to 3.

The OBDT of a Boolean function may be constructed from a truth table. Alternatively, OBDTs and OBDDs may be derived by repeatedly applying the Shannon expansion theorem (Equation (2.12)) to the algebraic description of a Boolean function. These structures can then be manipulated, using the reduction rules defined previously, to form canonical ROBDDs.
$f\left(x_{3}, X_{2}, x_{1}\right)$


Figure 2.5: (a) OBDT of the Boolean function, (b) OBDD, showing redundant nodes and isomorphic subfunctions, (c) ROBDD of the Boolean function.

The ROBDD is a canonical representation of a Boolean function and the number of nodes in the ROBDD cannot be reduced [31]. It is, however, possible to alter the order in which the function variables are encountered in the ROBDD. This leads to the construction of another ROBDD which is also a canonical representation of the original Boolean function. Hence, ROBDDs representing a Boolean function may be constructed where the number of nodes in each ROBDD varies, and is indeed dependent on the order of the variables $[30,31]$. The total number of OBDTs which may be constructed to represent a $n$ variable Boolean function is n!. Additionally, any $n$ variable Boolean function may also be represented by a total of $n$ ! ROBDDs.

In the BDTs, OBDDs and ROBDDs which have been introduced, each nonterminal node has been described as representing a function variable, $x_{\boldsymbol{J}}$ However, it is also possible to employ non-terminal nodes to represent subfunctions [30, 31]. This allows subfunction sharing and extends the use of the structures to the representation of multiple-output Boolean functions. Additionally, the OBDTs, OBDDs and ROBDDs have been considered as two-level representations of Boolean functions. If the root of the OBDT (OBDD, ROBDD) is assumed to correspond to the output of a circuit and the terminal nodes to the primary inputs then the structure can be considered to be a multiple-level representation of a circuit implementing a Boolean function. Indeed, the levels of the structures correspond to the levels of a multiple-level circuit. Hence, ROBDDs may be employed as concise multiple-level representations of multiple-output Boolean functions.

The concept of graphically representing Boolean functions may be further developed, resulting in structures which may be employed to describe FPRM expansions. These graphical constructions are called Reed-Muller Binary Decision Trees and are detailed fully in chapter 7.

### 2.7 Reed-Muller Universal Logic Module Networks

A Reed-Muller Universal Logic Module (RM-ULM) is a device which may be used to implement switching functions [3]. The device has control inputs, $2^{c}$ data inputs and a single output. The number of control inputs to a particular RM-ULM may be indicated by employing the notation RM-ULM(c). The symbol of the RM-ULM(c) is given in Figure 2.6.


Figure 2.6: Symbol of a RM-ULM(c)
The output of a RM-ULM $(c)$ is described by the switching function $f$,

$$
f=g_{0} \oplus g_{1} \dot{x}_{1} \oplus g_{2} \dot{x}_{2} \oplus g_{3} \dot{x}_{2} \dot{x}_{1} \oplus \ldots \ldots \oplus g_{2^{-1}-1} \dot{x}_{c} \dot{x}_{c-1} \ldots \dot{x}_{2} \dot{x}_{1}
$$

where $g_{f} \in\{0,1\}$ is a data input and $\dot{x}_{j}$ is a variable of the switching function $\left(\dot{x}_{j}=x_{j}\right.$ or $\left.\bar{x}_{j}\right) .\left(i=0,1, \ldots, 2^{c}-1, j \leq 1,2, \ldots, c\right)$

The circuits and symbols for RM-ULM(1) and RM-ULM(2) are illustrated in Figure 2.7. RM-ULM(1) is a single control input device ( $c=1$ ), and RM-ULM(2) is a device with 2 control inputs ( $c=2$ ), they implement the switching functions $f=g_{0} \oplus g_{1} \dot{x}_{1}$ and $f=g_{0} \oplus g_{1} \dot{x}_{1} \oplus g_{2} \dot{x}_{2} \oplus g_{3} \dot{x}_{2} \dot{x}_{1}$, respectively.

A RM-ULM(c) can directly implement any FPRM expansion of $c$ variables by employing the functional domain coefficients, $b_{i}\left(i=0,1, \ldots, 2^{n}-1\right)$, as the data inputs and the expansion variables, $x_{j}(j=1,2, \ldots, n)$, as the control inputs to the module. This is verified by observing that any FPRM expansion of $c$ variables may be represented by the expansion

$$
\begin{equation*}
f\left(x_{c}, x_{c-1}, \ldots, x_{1}\right)=b_{0} \oplus b_{1} \dot{x}_{1} \oplus b_{2} \dot{x}_{2} \oplus b_{3} \dot{x}_{2} \dot{x}_{1} \oplus \ldots \ldots \oplus b_{2-1} \dot{x}_{c} \dot{x}_{c-1} \ldots \dot{x}_{2} \dot{x}_{1} \tag{2.25}
\end{equation*}
$$

$b_{1} \in\{0,1\}, \quad i=0,1, \ldots, 2^{c}-1$.
Equation (2.25) is identical to Equation (2.24), the output of a RM-ULM(c), when $b_{i}$ is equated with $g_{i}$.


Figure 2.7: Circuits and symbols of (a) RM-ULM(1) and (b) RM-ULM(2)

It is also possible to utilise a RM-ULM( $c$ ) to implement any FPRM expansion of $(c+1)$ variables. This is now demonstrated,
A FPRM expansion of $(c+1)$ variables

$$
\begin{aligned}
f\left(x_{c+1}, x_{c} \ldots, x_{1}\right)= & b_{0} \oplus b_{1} \dot{x}_{1} \oplus b_{2} \dot{x}_{2} \oplus b_{3} \dot{x}_{2} \dot{x}_{1} \oplus \ldots \ldots \oplus b_{2^{c-1}} \dot{x}_{c} \dot{x}_{c-1} \ldots \dot{x}_{2} \dot{x}_{1} \oplus b_{2} \dot{x}_{c+1} \\
& -b_{2^{+}+1} \dot{x}_{c+1} \dot{x}_{1} \oplus b_{2^{c+2}} \dot{x}_{c+1} \dot{x}_{2} \oplus b_{2^{+3}} \dot{x}_{c+1} \dot{x}_{2} \dot{x}_{1} \oplus \ldots \ldots . \oplus b_{2^{c+1}-1} \dot{x}_{c+1} \dot{x}_{c} \ldots \dot{x}_{2} \dot{x}_{1} \\
= & \left(b_{0} \oplus b_{2^{2} \dot{x}_{c+1}}\right) \oplus\left(b_{1} \oplus b_{2^{+}+1} \dot{x}_{c+1}\right) \dot{x}_{1} \oplus\left(b_{2} \oplus b_{2^{+}+2} \dot{x}_{c+1}\right) \dot{x}_{2} \oplus \ldots \\
& \ldots \oplus\left(b_{2^{c}-1} \oplus b_{2^{c+1}-1} \dot{x}_{c+1}\right) \dot{x}_{c} \ldots \dot{x}_{2} \dot{x}_{1}
\end{aligned}
$$

Equating

$$
B_{i}=b_{i} \oplus b_{2^{+}+1} \dot{x}_{c+1}= \begin{cases}0 & \text { if } b_{1}=0, b_{2^{+}+1}=0 ; \\ 1 & \text { if } b_{i}=1, b_{2^{+}+1}=0 ; \\ \dot{x}_{c+1} & \text { if } b_{1}=0, b_{2^{+}+1}=1 ; \\ \bar{x}_{c+1} & \text { if } b_{i}=1, b_{2^{+}+1}=1\end{cases}
$$

$$
i=0,1,2, \ldots, 2^{c}-1 .
$$

then

$$
\begin{equation*}
f\left(x_{c+1}, x_{c} \ldots, \ldots x_{1}\right)=B_{0} \oplus B_{1} \dot{x}_{1} \oplus B_{2} \dot{x}_{2} \oplus B_{3} \dot{x}_{2} \dot{x}_{1} \oplus \ldots \ldots B_{2-1} \dot{x}_{c} \dot{x}_{c-1} \ldots \dot{x}_{2} \dot{x}_{1} \tag{2.26}
\end{equation*}
$$

Equation (2.26) is identical to (2.24), the output of a RM-ULM(c), when $B_{j}$ is equated with $g_{j}$ Hence, a RM-ULM(c) can implement any FPRM expansion of ( $c+1$ ) variables, where each data inputs of the $\operatorname{RM}-\operatorname{ULM}(c)$ may be 0,1 , $x_{j}$ or $\bar{x}_{j}$. This technique is illustrated by using a RM-ULM(2) to implement a function of 3 variables.
FPRM expansion ( $n=3$ )

$$
\begin{equation*}
f\left(x_{3}, x_{2}, x_{1}\right)=b_{0} \oplus b_{1} \dot{x}_{1} \oplus b_{2} \dot{x}_{2} \oplus b_{3} \dot{x}_{2} \dot{x}_{1} \oplus b_{4} \dot{x}_{3} \oplus b_{3} \dot{x}_{3} \dot{x}_{1} \oplus b_{6} \dot{x}_{3} \dot{x}_{2} \oplus b_{7} \dot{x}_{3} \dot{x}_{2} \dot{x}_{1} \tag{2.27}
\end{equation*}
$$

Rearranging

$$
\begin{aligned}
f\left(x_{3}, x_{2}, x_{1}\right) & =\left(b_{0} \oplus b_{4} \dot{x}_{3}\right) \oplus\left(b_{1} \oplus b_{3} \dot{x}_{3}\right) \dot{x}_{1} \oplus\left(b_{2} x \oplus b_{6} \dot{x}_{3}\right) \dot{x}_{2} \oplus\left(b_{3} \oplus b_{7} \dot{x}_{3}\right) \dot{x}_{2} \dot{x}_{1} \\
& =B_{0} \oplus B_{1} \dot{x}_{1} \oplus B_{2} \dot{x}_{2} \oplus B_{3} \dot{x}_{2} \dot{x}_{1}
\end{aligned}
$$

$$
\begin{aligned}
& B_{0}=b_{0} \oplus b_{4} \dot{x}_{3} \\
& B_{1}=b_{1} \oplus b_{5} \dot{x}_{3} \\
& B_{2}=b_{2} \oplus b_{6} \dot{x}_{3} \\
& B_{3}=b_{3} \oplus b_{7} \dot{x}_{3}
\end{aligned}
$$

where $b_{i} \in\{0,1\}$, and $B_{i} \in\left\{0,1, \dot{x}_{3}, \bar{x}_{3}\right\}$
Hence, $B_{i}(i=0,1,2,3)$ are the data input of the RM-ULM(2), as illustrated in Figure 2.8. The output of this module is

$$
f=B_{0} \oplus B_{1} \dot{x}_{1} \oplus B_{2} \dot{x}_{2} \oplus B_{3} \dot{x}_{2} \dot{x}_{1}
$$

A RM-ULM network may be formed by connecting the data inputs of one RM-ULM to the outputs of other RM-ULMs. Indeed, each data input of RMULM (c) may be connected to the output of another module, RM-ULM( $d$ ), to form a network which consists of $2^{c} \mathrm{RM}-\mathrm{ULM}(d) \mathrm{s}$ and a single RM-ULM(c). This network is equivalent to a single RM-ULM $(c+d)$. This is now detailed, assuming that the data inputs of $2^{c}$ RM-ULM $(d)$ s are $g_{k}\left(k=0,1, \ldots .2^{c+d}-1\right)$, and the control inputs are $\dot{x}_{1}, \dot{x}_{2}, \ldots, \dot{x}_{d}$ as illustrated in Figure 2.9.


Figure 2.8: RM-ULM(2) implementation of the FPRM expansion given in Equation (2.27)

The output of each RM-ULM(d) is given by:

$$
\begin{aligned}
& f^{0}=g_{0} \oplus g_{1} \dot{x}_{1} \oplus g_{2} \dot{x}_{2} \oplus g_{3} \dot{x}_{2} \dot{x}_{1} \oplus \ldots \ldots \oplus g_{2-1} \dot{x}_{d} \dot{x}_{d-1} \ldots \dot{x}_{2} \dot{x}_{1} \\
& f^{1}=g_{2^{\alpha}} \oplus g_{2^{4+1}} \dot{x}_{1} \oplus g_{2^{\alpha}+2} \dot{x}_{2} \oplus g_{2^{\alpha+3}} \dot{x}_{2} \dot{x}_{1} \oplus \ldots \ldots \oplus g_{2^{\alpha+1}-1} \dot{x}_{d} \dot{x}_{d-1} \cdot \dot{x}_{2} \dot{x}_{1} \\
& f^{2^{x-1}}=g_{2^{-\alpha-1}} \oplus g_{2^{+\alpha-1}+1} \dot{x}_{1} \oplus g_{2^{x-d-1}+2} \dot{x}_{2} \oplus g_{2^{x-1-1}+3} \dot{x}_{2} \dot{x}_{1} \oplus \ldots . . . \oplus g_{2^{x-\alpha-1}} \dot{x}_{d} \dot{x}_{d-1} \cdots \dot{x}_{2} \dot{x}_{1}
\end{aligned}
$$

where each $f^{i}\left(i=0,1, \ldots, 2^{c}-1\right)$ is the output of the $i$ th $\operatorname{RM}-\operatorname{ULM}(d)$ in the network and is a data input of the first level module RM-ULM(c), as illustrated in Figure 2.9. If $\dot{x}_{d+1}, \dot{x}_{d+2}, \ldots, \dot{x}_{c+d}$ are the control inputs of the module RM-ULM(c), then the output $f$ is given by

$$
\begin{align*}
& f=f^{0} \oplus f^{1} \dot{x}_{d+1} \oplus f^{2} \dot{x}_{d+2} \oplus f^{3} \dot{x}_{d+2} \dot{x}_{d+1} \oplus \ldots \ldots \oplus f^{2-1} \dot{x}_{c+d} \dot{x}_{c+d-1} \ldots \dot{x}_{d+2} \dot{x}_{d+1} \\
& =g_{0} \oplus g_{1} \dot{x}_{1} \oplus g_{2} \dot{x}_{2} \oplus \ldots . . . \oplus g_{2-1}{ }^{-1} \dot{x}_{d} \dot{x}_{d-1} \cdots \dot{x}_{2} \dot{x}_{1} \oplus g_{2} \dot{x}_{d+1} \\
& \oplus g_{2^{d}+1} \dot{x}_{d+1} \dot{x}_{1} \oplus g_{2^{d}+2^{x_{d+1}}} \dot{x}_{2} \oplus \ldots . . . \oplus g_{2^{-1+1}-1} \dot{x}_{d+1} \dot{x}_{d} \dot{x}_{d-1} \ldots \dot{x}_{2} \dot{x}_{1} \oplus \ldots . . \\
& \oplus 8_{2^{-\alpha-1}} \dot{x}_{c+4} \dot{x}_{d+1} \oplus 8_{2-\alpha-1+1} \dot{x}_{c+4} \cdot \dot{x}_{d+1} \dot{x}_{1} \oplus 8_{2^{-\alpha-1}+2} \dot{x}_{c+4} \cdot \dot{x}_{d+1} \dot{x}_{2} \oplus \ldots . . \\
& \oplus g_{2^{2+-4}-1} x_{c+d} d_{c+d-1} \cdots t_{d+1} x_{d} \dot{x}_{d-1} \cdots t_{2} x_{1} \tag{2.28}
\end{align*}
$$



The output of a RM-ULM $(c+d)$ is

$$
\begin{equation*}
f(c+d)=g_{0} \oplus g_{1} \dot{x}_{1} \oplus g_{2} \dot{x}_{2} \oplus g_{3} \dot{x}_{2} \dot{x}_{1} \oplus \ldots \ldots \oplus g_{2-d-1} \dot{x}_{c+d} \dot{x}_{c+d-1} \cdots \dot{x}_{2} \dot{x}_{1} \tag{2.29}
\end{equation*}
$$

Equations (2.28) and (2.29) are identical, hence implementing a network using $2^{c}$ RM-ULM $(d) S$ and a single RM-ULM(c) is equivalent to an implementation comprised of a single RM-ULM(c+ $d$ ).

Any data input of a RM-ULM may be the binary value 0 or 1 , literal $x_{j}$ or $\bar{x}_{j}$ or the output of another RM-ULM. A tree network is a network of 3 or more RM-ULMs. Additionally, 2 or more data inputs of at least one RM-ULM in the network must be the outputs of other RM-ULMs in the network. An example of a tree network is illustrated in Figure 2.10, implementing the FPRM expansion of Equation (2.30).

$$
\begin{equation*}
f\left(x_{5}, x_{4}, x_{3}, x_{2}, x_{1}\right)=\bar{x}_{3} x_{1} \oplus \bar{x}_{3} x_{2} \oplus x_{4} \oplus \bar{x}_{3} \oplus \bar{x}_{5} x_{4} \bar{x}_{3} \tag{2.30}
\end{equation*}
$$



Figure 2.10: Tree network implementing the FPRM expansion given in Equation (2.30).

A cascade network is formed from a minimum of 2 RM-ULMs where a maximum of one data input of each RM-ULM is connected to the output of another RM-ULM in the network. An example of a cascade network is illustrated in Figure 2.11, implementing the FPRM expansion of Equation (2.31).

$$
\begin{equation*}
f\left(x_{5}, x_{4}, x_{3}, x_{2}, x_{1}\right)=x_{1} \oplus x_{2} \oplus \bar{x}_{3} \oplus x_{4} \oplus \bar{x}_{5} \tag{2.31}
\end{equation*}
$$



Figure 2.11: Cascade network implementing the FPRM expansion given in Equation (2.31).

### 2.8 Summary

The preceding sections of this chapter have reviewed some principles of switching theory. This has included comprehensive descriptions and definitions of logic functions, various types of RM expansions and ESOP forms. Additionally, the theory of BDDs and RM-ULMs has been discussed. Hence, this chapter supports the research work detailed in the remainder of this thesis.

# Chapter 3 <br> Logic Minimisation Using Fixed Polarity Reed-Muller Expansions' 

An arbitrary $n$ variable switching function may be represented as an exclusive-OR sum-of-products form. This has been demonstrated in chapter 2. Constraints can be imposed which limit each variable to appearing in either true or complemented form throughout an ESOP expression. This gives rise to the $2^{n}$ fixed polarity Reed-Muller expansions formally defined in section 2.3 of chapter 2 . The numbers of product terms and literals in each FPRM expansion will vary, depending on the form of the original switching function. Hence, some FPRM expansions may be more efficient representations than other FPRM forms.

There are many criteria for determining what constitutes an optimum representation of a switching function. One possible criterion may be that of minimisation, hence the optimum representations may be those which are comprised of the fewest numbers of product terms and literals. This may lead to reductions in the number of components and overall area required to implement the original function. Alternatively, the emphasis may be placed on deriving expressions which are easily tested. A third criterion may be that of minimising the timing delays through the final implementation. It is possible, however, that the optimum representation is that which meets several criteria. Hence, it may be necessary to determine representations which are judged to be good, though not necessarily optimum, when individual parameters are assessed.

The criterion for determining the efficiency of FPRM representations of switching functions which is adopted in this thesis is that of minimisation. The parameter used for assessment is the number of product terms in a FPRM expansion and this is termed the weight of the expansion. The FPRM expansions which contain fewest product terms, and hence have the lowest

[^0]weight, are deemed to be optimum solutions. It may be argued that the EXOR operator is more expensive to implement, in terms of area and speed, than the AND and invert operators. This may be overcome by introducing cost functions, where the EXOR operator has a higher value than the AND and invert operators. The scheme adopted in this thesis, that is, to count the number of product terms in an expansion without applying cost functions, is both simple and commonly used. It has the additional advantage of being technology independent.

Determining the optimum FPRM expansion of a Boolean function is a considerable task as each $n$ variable function may be represented by $2^{n}$ FPRM expansions. It is possible to exhaustively search for the minimum expansion, however, as the number of variables increases the time and memory allocations required for the search become impractical. The following section of this chapter reviews techniques for representing, deriving and minimising ESOP expressions with a particular emphasis being placed on techniques for FPRM expansions. A heuristic minimisation technique for determining minimal FPRM expansions, based on an algorithm developed by Marinkovic and Tosic [21], is presented in section 3.2. The switching functions are represented using a method devised by Almaini, Thomson and Hanson [1]. The performance of the algorithm is evaluated and results are presented.

### 3.1 Review of Techniques for Fixed Polarity Reed-Muller Expansions

The purpose of this literature review is to summarise some of the many techniques which are available for representing, generating and optimising FPRM expansions. The techniques reviewed include those commonly implemented and tested and also those which are particularly relevant to the work presented in this chapter.

Muller [24] and Cohn [28] initially employed algebraic equations to represent ESOP forms, where each operator (AND, EXOR) and operand were explicitly denoted. This type of representation can result in unwieldy and inefficient expressions, hence more compact methods of describing FPRM expansions and ESOP forms have been devised. Most forms of representation and, indeed, many methods for generating and minimising FPRM expansions are extensions of techniques employed in the Boolean domain.

Perhaps Karnaugh maps provide one of the most familiar methods of representing Boolean functions. This type of representation supports both completely and incompletely specified functions. Additionally, a Boolean function represented by a Karnaugh map may be minimised according to a set of pre-defined rules. However, the Karnaugh map method has two main limitations. Firstly, maps becomes unwieldy and difficult to visualise when the number of variables in the Boolean functions increase. Secondly, the minimisation procedure, which involves forming groups of minterms, is to some extent intuitive relying on the ability and experience of the user. This second limitation leads to difficulties when trying to automate the minimisation process. The counterpart of the Karnaugh map is the ReedMuller coefficient map [35], a technique for representing FPRM expansions. Wu, Chen and Hurst [35] initially demonstrated the representation of the PPRM expansion. Each coefficient $a_{i}\left(i=0,1, \ldots, 2^{n}-1\right)$ of the expression is plotted in a cell of the map, the relationship between cells being EXOR or modulo-2 addition. Wu et al also formulated a procedure whereby plotting the $d_{1}$ coefficients of a Boolean function resulted in a map representation of the equivalent PPRM expansion. The RM coefficient maps of each FPRM expansion may be generated from the map representing the PPRM expansion. This is achieved through a 'folding' operation performed on the RM coefficient map where modulo-2 addition is carried out between the contents of certain cells of the map. A heuristic minimisation technique has been developed from this procedure which determines the optimal (suboptimal) equivalent FPRM expansion of an initial Boolean function. Additionally, Wu et al considered deriving minimal (sub-minimal) ESOP forms from the RM coefficient map. This technique, similar in principle to minimising Boolean functions using Karnaugh maps, is more fully detailed in chapter 5.

Tran [36] demonstrated that it is possible to generate the equivalent RM coefficient map of any FPRM expansion from the Karnaugh map representing the Boolean function. This technique employs an adapted form of the folding operation developed in [35]. The number of folds which must be made is equal to $n$, the number of variables present in the Boolean function and the only arithmetic operation undertaken is modulo-2 addition. The RM coefficient map was also extended to allow the representation of the FPRM expansions of incompletely specified Boolean functions. Again, the folding
technique is employed to derive the $R M$ coefficient map of a FPRM expansion from an incompletely specified Boolean function initially portrayed on a Karnaugh map. It is then possible to assign values to the 'don't care' terms with the goal of minimising the number of cells of the RM coefficient map which contain the value 1. Tran described a heuristic method which relies on the judgement of the user. Green [37] also used the map method to assign the 'don't care' terms of incompletely specified functions. The procedure involves transforming all specified terms of a Boolean switching function to the RM domain. All combinations of unspecified terms are then separately transformed and 'added' to the terms on the RM coefficient map of completely specified terms. The map of unspecified terms which provides the greatest reduction in the number of terms on the RM coefficient map provides the optimum solution. This method exhaustively searches for the optimum assignment of 'don't care' terms. The use of RM coefficient maps is further extended as Green considered jointly the best use of 'don't care' terms and the optimum polarity.

Tri-state maps [38] are similar in structure to Karnaugh and RM coefficient maps. However, this new representation has one attribute which makes it valuable when used in the process for converting Boolean functions to FPRM expansions. Throughout the conversion process the polarity of each variable is clearly indicated by the tri-state map. Thus, RM coefficient maps may be modified to adopt this characteristic of tri-state maps. The folding technique described previously may once again be employed as the technique for converting the maps from one representation to another. In addition to this work $\operatorname{Tran}[36,38]$ considered the minimisation of ESOP forms using RM coefficient maps, this is reviewed in detail in chapter 5.

The technique of map-compression, that is, using map-entered variables, is often applied to Boolean functions represented on Karnaugh maps. This technique has been extended to the RM domain by Green [37], making it possible to compress RM coefficients maps and thus reduce the number of folds required when generating the RM coefficient map representing a FPRM expansion. The technique, when applied to Karnaugh maps reduces the number of folds needed to obtain the equivalent $R M$ coefficient map representation of a function.

Besslich [39] utilised a signal flow diagram to generate any FPRM expansion from an initial Boolean function. Each diagram is constructed according to the relationship between the $d_{i}$ coefficients of the $n$ variable Boolean function and the $b_{k}$ coefficients of the final FPRM expansion ( $i, k=$ $\left.0,1, \ldots, 2^{n}-1\right)$. Thus, the inputs to the signal flow diagram are the $d_{1}$ coefficients whilst the outputs are the $b_{k}$ coefficients. The conversion employs only modulo-2 addition and the number of operations which must be undertaken is $n 2^{n-1}$. Additionally, Besslich addressed the problem of determining the optimum FPRM expansion of a Boolean function by devising an efficient exhaustive technique. A FPRM expansion may be derived from another FPRM expansion by complementing any single expansion variable throughout the initial expression. If the order in which the FPRM expansions are generated is Gray code ordering then it is possible to sequentially construct all $2^{n}$ FPRM expansions. Besslich performed this task using signal flow diagrams, transforming one FPRM expansion to another FPRM expansion where the polarity of a single expansion variable is complemented. This operation requires $2^{n-1}$ modulo-2 additions. Hence, deriving all $2^{n}$ FPRM from an initial Boolean function would require a total of $\left(n 2^{n-1}+\left(2^{n}-1\right) 2^{n-1}\right)$ modulo-2 additions. The optimum FPRM expansion may then be selected from the complete set of all $2^{n}$ expressions.

The transform triangle [3, 4] is an alternative graphical technique for deriving the equivalent RM expansions of Boolean functions initially expressed in minterm form. This method involves listing all $d_{i}$ coefficients ( $i=0,1, \ldots, 2^{n}-1$ ) of the Boolean function as the first row of a triangular array. Each consecutive pair of coefficients are then summed using modulo2 addition, the results form the next row of the array. Each consecutive pair of these new coefficients are then summed modulo-2 and the third row of the array is formed. This is repeated until a row is formed which comprises of only a single coefficient. The $a_{i}$ coefficients of the RM expansion are listed down the left-hand edge of the triangle where coefficient $a_{0}$ occupies the first row, $a_{1}$ occupies the second row and so on. If this technique is employed to generate the RM expansion of a $n$ variable Boolean function then the number of modulo-2 additions which must be executed is $\sum_{i=1}^{i=2^{2}-1} i$.

Marinkovic and Tosic [21] introduced both an exhaustive and a heuristic
technique for deriving minimal (sub-minimal) FPRM expansions. Both techniques are iterative and commence by evaluating the number of literals in an arbitrarily selected FPRM expansion. The state of a single expansion variable is complemented during each iteration. The variable to be complemented is chosen by evaluating the effects of complementing each variable in turn. The variable which causes the greatest reduction in the number of literals in the FPRM expansion is then complemented resulting in a new FPRM expansion. The technique is repeatedly applied to each new FPRM expansion until the number of literals cannot be further reduced. At this point the heuristic minimisation technique determines that this FPRM expansion is the minimal (sub-minimal) expression. However, the exhaustive technique continues by selecting another FPRM expansion which has not previously been derived or, indeed, evaluated. If the number of literals in this expression can be reduced then the appropriate variable is complemented and the algorithm is repeatedly applied until no further reduction is possible. This is done until the number of literals in all $2^{n}$ FPRM expansions has been calculated. It should be noted that in order to perform this task it is not necessary to derive all FPRM expansions, it is only necessary to determine the number of literals in each expression. These minimisation techniques are further considered in the following sections of this chapter.

The tabular technique [1] may be employed to convert any completely specified Boolean function to an equivalent FPRM expansion. This aptly named technique employs a tabular representation where each minterm of a Boolean function or product term of a FPRM expansion is explicitly denoted in binary form. Additionally, a heuristic algorithm for deriving minimal (sub-minimal) FPRM expansions has been developed. This iterative method does not guarantee to find the optimum FPRM expansion but at each iteration reduces the number of product terms in the representation. The tabular technique is more fully described in section 3.2.1.

Habib [40, 41] developed exhaustive and heuristic algorithms for deriving FPRM expansions from Boolean functions. The techniques utilise a Boolean matrix representation where the $d_{i}\left(i=0,1, \ldots, 2^{n}-1\right)$ coefficients of the $n$ variable Boolean function are represented in a $1 \times 2^{n}$ Boolean matrix. The matrix undergoes a series of $n$ transformations, similar to folding a

Karnaugh map. This operation does not employ matrix multiplication, the only arithmetic operation being that of modulo- 2 addition. The resulting matrix represents a FPRM expansion of the function where the polarity of the expression is decided by the user. Hence, this technique may be employed to sequentially generate each FPRM expansion of a Boolean function making it possible to determine the optimum equivalent FPRM expansion through exhaustive search. Additionally, Habib [41] adapted this technique to form a heuristic method for determining the minimal (subminimal) FPRM expansion of a Boolean function. These techniques may also be employed to generate FPRM expansions from incompletely specified Boolean functions as the 'don't care' terms are assigned so as to match their corresponding term in the partitioned matrix. This is, however, a heuristic method and it cannot be guaranteed that the 'don't care' terms have been optimally assigned or that the minimal FPRM expansion has been determined. The algorithms developed by Habib include another useful technique, that of identifying independent variables. This is implemented whilst converting a Boolean function to the equivalent FPRM expansion and, hence, does not add any overheads to the conversion procedure. Finally, the nature of the Boolean matrix representation and the fact that the minimisation procedures are not intuitive make these algorithms suitable for automation.

Harking [42] presented a novel algorithm which may be employed to form a polarity matrix denoting all $2^{n}$ FPRM expansions of any Boolean function. Initially, the coefficients $d_{1}\left(i=0,1, \ldots, 2^{n}-1\right)$ of the $n$ variable Boolean function are represented by a Boolean matrix of dimension $1 x 2^{n}$. A $2^{n} \times 2^{n}$ Boolean matrix is then iteratively constructed. Each row of this matrix denotes the $b_{i}\left(i=0,1, \ldots, 2^{n}-1\right)$ coefficients of the polarity $p(p=0,1, \ldots$, $2^{n}-1$ ) FPRM expansion of the original Boolean function. The matrix is formed through modulo-2 addition, no matrix multiplication is undertaken. Harking also describes a modified form of this algorithm where any single FPRM expansion may be constructed without forming the $2^{n} \times 2^{n}$ polarity matrix. This technique is suitable when $n$ is large and it is impractical to calculate a matrix with dimensions $2^{n} \times 2^{n}$. Additionally, the generation of FPRM expansions from incompletely specified Boolean functions is discussed. Three procedures were detailed for deriving optimal (sub-optimal) FPRM expansions, one of these methods is exhaustive whilst the remaining
techniques are heuristic.

The representation of switching function using the operators of modulo-2 algebra makes it possible to employ the mathematical devices which support this algebra. One area where this has been exploited is in using transform matrices to convert Boolean functions to FPRM expansions [43]. The $d_{i}(i$ $=0,1, \ldots, 2^{n}-1$ ) coefficients of the $n$ variable Boolean function are represented by a $2^{n} \times 1$ Boolean matrix which is multiplied by the $2^{n} \times 2^{n}$ Reed-Muller transform matrix. This yields a new $2^{n} x 1$ Boolean matrix representing the $a_{1}$ coefficients of the equivalent PPRM expansion. Modulo-2 addition and modulo-2 multiplication are employed throughout. The recursive structure of the RM transform matrix makes it possible to redefine this matrix using the Kronecker product. Hence, any $2^{n} \times 2^{n} R M$ transform matrix may be constructed from a basic $2 x 2$ matrix. It is also interesting to note that multiplying the matrix representing any PPRM expansion by the RM transform matrix will result in the matrix representing the equivalent Boolean function. That is, the RM transform matrix is its own inverse. Transform matrices may be employed to generate all FPRM expansions from the positive polarity form or indeed the initial Boolean function [37]. Two different transform matrices are applied according to the polarity of each variable in the final FPRM expansion. Each new FPRM expansion may be generated from the previous one by a single matrix transformation if Gray code order is used to determine the sequence in which FPRM expansion should be derived.

The techniques developed by Zhang and Rayner [44] may be employed to efficiently derive FPRM expansions of Boolean functions. The $2^{n} \times 2^{n} R M$ transform matrix, as described previously, is employed to transform Boolean functions to FPRM expansions. The coefficients of these FPRM expansions and Boolean functions are represented using $2^{n} \times 1$ Boolean matrices and matrix multiplication is performed during the conversion from the operational domain to the functional domain. Zhang and Rayner described a means for reducing the number of modulo-2 additions which must be performed when generating any FPRM expansion from a $n$ variable Boolean function. This is achieved through factoring the RM transform matrix into $n$ matrices using the Kronecker product form. The resulting technique is called the Fast Reed-Muller Transform (FRMT) algorithm. Thus, the number
of modulo-2 additions which must be performed when generating a FPRM expansion from a $n$ variable Boolean function has been reduced from ( $3^{n}-2^{n}$ ) to $n 2^{n-1}$. Additionally, Zhang and Rayner illustrated a technique for deriving each FPRM expansion from an existing FPRM expression of different polarity using only a single matrix from the Kronecker form of the RM transform matrix. Gray code ordering was used to determine the sequence in which the FPRM expansions should be derived, i.e. which new FPRM expansion should be generated from the existing expression. Thus all FPRM expansions may be generated sequentially from an initial Boolean function. The total number of modulo-2 operations which must be performed is $\left(n 2^{n-1}+\left(2^{n-1}\right)\left(2^{n-1}\right)\right)$. This technique was explored by Green [37], who also demonstrated the derivation of all $2^{n}$ FPRM expansions, represented by a $2^{n} x 2^{n}$ Boolean matrix. The minimal FPRM expansion may then be determined by locating the row of the matrix comprised of the fewest number of 1 s . Saluja and Ong [45] also employed the RM transform matrix to generate all FPRM expansions of a Boolean function. Thus the optimum FPRM expansion of a Boolean function may be found through exhaustive search. This method differs from the transform matrix techniques discussed previously in that only the RM transform matrix is employed and the Boolean matrix representing the $d_{i}$ coefficients of the Boolean function is repeatedly modified to generate each new FPRM expansion. This technique does, however, employ matrix multiplication which becomes inefficient for Boolean functions with large numbers of variables.

Sarabi and Perkowski [46] described a technique for deriving FPRM expansions from Boolean functions expressed in disjoint SOP form. Additionally, two minimisation algorithms were presented, one exact, the other heuristic. Both techniques operate by considering the state of each variable in the Boolean function and altering the polarity of the variable according to certain rules. Results are presented which indicate the performance of the heuristic algorithm when compared to existing heuristic algorithms. The algorithms are tested using benchmark functions and the results are referenced again in section 3.2.2 of this chapter.

Purwar [47] developed a novel technique for deriving FPRM expansions from a Boolean function initially represented by a BDD. This method differs from many of the techniques already discussed in this literature review in
that the initial Boolean representation may be a disjoint SOP form, i.e. the product terms need not be minterms. Purwar determined each coefficient, $a_{j},\left(j=0,1, \ldots, 2^{n}-1\right)$ of the equivalent RM expansion by considering the contribution of each relevant path through the BDD. Only paths which terminate in the Boolean constant 1 need be evaluated. The $b_{j}$ coefficients of any FPRM expansion may also be determined from the BDD. The efficiency of this technique is determined by the number of paths of the BDD which must be evaluated. If the number of paths which terminate in the constant 1 is greater than the number of paths which terminate in constant 0 then Purwar concluded that the complement of the FPRM expansion be derived through evaluating paths which terminate in the value 0 . The nature of FPRM expansions means that the task of recomplementing the expansion is a trivial one. This method of generating FPRM expansions is particularly useful as the initial representation may be a disjoint Boolean SOP form.

Functional Decision Diagrams (FDDs) [48, 49] provide a graphical form of representation for FPRM expansions and are similar in structure to BDDs used to represent Boolean functions. Kebschull, Schubert and Rosenstiel [48] described an algorithm for deriving the factored forms of FPRM expansions from the FDDs representing the expressions. Hence, FDDs provide multi-level representations of FPRM expansions. The structure and uses of FDDs are further investigated in chapter 7.

Miller and Thomson [50] described an efficient technique for determining minimal FPRM expansions of Boolean functions through exhaustive search. The technique may be modified so as to utilise a heuristic technique to isolate a sub-minimal FPRM expansion which may then be used as a starting expression for the subsequent search. Additionally, a non-exhaustive method of determining the optimum FPRM expansions of any 3 variable Boolean function was presented.

Other techniques for deriving minimal (sub-minimal) FPRM expansions of Boolean functions include those developed by Davio, Deschamps and Thayse [22], Mukhopadhyay and Schmitz [51], Lui and Muzio [52], Ungern [53] and Clarkson and Zhuang [54]. Also of interest is work presented by Sasoa [7], who considered the numbers of product terms in the FPRM expansions of
different types of Boolean function and also compares the efficiency of FPRM representations with different categories of ESOP forms. Additionally, Csanky, Perkowski and Schafer [17, 55] presented a heuristic algorithm for deriving minimal (sub-minimal) canonical restricted mixed polarity forms of Boolean functions. These forms are comprised of all FPRM expansions and the inconsistent canonical forms identified by Cohn [28].

Techniques developed by Lui and Muzio [52, 56, 57, 58], Falkowski and Perkowski [59, 60], Riege and Besslich [61] and Varma and Trachtenberg [62] may also be considered to be related to the work presented in the following sections of this chapter. However, the techniques presented in these publications are more relevant to the work described in subsequent chapters of this thesis. Hence, reviews are undertaken in the appropriate chapters.

### 3.2 Minimisation Techniques for Fixed Polarity Reed-Muller Expansions

A variety of techniques for representing and deriving FPRM expansions have been reviewed in the previous section. Additionally, algorithms for determining minimal (sub=minimal) FPRM expansion have been examined. The work presented in this section documents the evolution of a heuristic minimisation algorithm. The algorithms on which this new technique is based are first reviewed and their strengths and weaknesses discussed. The goal of this work is to formulate a technique which shows improvements in both efficiency and the quality of the solution obtained when evaluated against existing techniques.

### 3.2.1 Review of Tabular Techniques

The tabular technique [1] considered in section 3.1 provides a means of converting a Boolean function to any FPRM expansion. The technique employs a notation whereby each term (minterm ( $m_{i}$ ), piterm ( $\pi_{i}$ ) or product term $\left.\left(\rho_{i}\right), i=0,1, \ldots, 2^{n}-1\right)$ of a $n$ variable expansion is represented as a binary $n$-tuple. Thus, the $i$ th term of an expansion is represented by the binary equivalent of $i$, that is, $\left\langle i_{n} i_{n-1} \ldots i_{1}\right\rangle, i_{j} \in\{0,1\}, j=1,2, \ldots, n$. Minterms, piterms and product terms have been defined and referring to the convention adopted in chapter 2 , it is possible to relate the state of the variables comprising any term to the notation employed by the tabular technique. Thus, if a minterm of a Boolean function contains the literal $\bar{x}_{j}$
$\left(x_{j}\right)$, then the binary $n$-tuple representing the minterm will contain the integer 0 (1) in position $j$. If a product term of a FPRM expansion is independent of literal $\dot{x}_{j}$ (contains literal $\dot{x}_{j}$ ) then the binary $n$-tuple representing the product term will contain the integer 0 (1) in position $j$.
e.g. for $n=3$,
$m_{3}$ is a minterm of a 3 variable Boolean function.

$$
m_{3}=\bar{x}_{3} x_{2} x_{1}=x_{3}^{0} x_{2}^{1} x_{1}^{1}=011
$$

$\pi_{3}$ is a piterm of a 3 variable RM expansion.
$n_{3}=x_{2} x_{1}=x_{3}^{0} x_{2}^{1} x_{1}^{1}=011$
$\rho_{5}$ is a product term of a 3 variable FPRM expansion.
$\rho_{5}=\dot{x}_{3} \dot{x}_{1}=\dot{x}_{3}^{1} \dot{x}_{2}^{0} \dot{x}_{1}^{1}=101$

Boolean functions, RM expansions and FPRM expansions may be represented by a list of terms where each column of the list represents a variable ( $\dot{x}_{j}$ ) and each row represents a term ( $m_{i}, n_{i}, p_{i}$ ) of the Boolean function, RM expansion or FPRM expansion. The number of columns is fixed and is equal to $n$. The number of rows is equal to the number of terms in the expansion. The following example illustrates the tabular representations of a Boolean function, RM expansion and FPRM expansion.

Example 3.1 Using the tabular notation display the following 3 variable Boolean function, RM expansion and FPRM expansion.

Boolean function

$$
\left.\begin{array}{rl}
f\left(x_{3}, x_{2}, x_{1}\right)= & \bar{x}_{3} \bar{x}_{2} \bar{x}_{1}+\bar{x}_{3} \bar{x}_{2} x_{1}+x_{3} \bar{x}_{2} x_{1}+x_{3} x_{2} x_{1} \\
\\
x_{3} & x_{2} \\
\hline 0 & 0 \\
1 \\
0 & 0 \\
1 \\
1 & 0 \\
1 & 1 \\
1 & 1
\end{array}\right) 1 .
$$

RM expansion

$$
\begin{aligned}
& f\left(x_{3}, x_{2}, x_{1}\right)=1 \oplus x_{2} \oplus x_{3} \oplus x_{3} x_{1} \oplus x_{3} x_{2} \\
& 0 \quad 1 \quad 0 \\
& 100 \\
& 10 \quad 1 \\
& 1 \quad 10
\end{aligned}
$$

FPRM expansion (Polarity 5)

$$
\begin{aligned}
& f_{5}\left(x_{3}, x_{2}, x_{1}\right)=1 \oplus \bar{x}_{1} \oplus \bar{x}_{3} \bar{x}_{1} \oplus \bar{x}_{3} x_{2} \\
& \bar{x}_{3} \\
& \hline x_{2}
\end{aligned} \bar{x}_{1} .
$$

The transformation of a Boolean function to the equivalent RM expansion is based on the equality

$$
\bar{x}_{j}=x_{j} \oplus 1
$$

The tabular technique for converting a Boolean function to a RM expansion is dependent on this equality. It is, therefore, necessary that it be realised in a form suitable for use with the notation described previously. Thus the equality may be expanded and expressed as

$$
\dot{x}_{n} \ldots \bar{x}_{j} \ldots \dot{x}_{1}=\dot{x}_{n} \ldots x_{j} \ldots \dot{x}_{1} \odot \dot{x}_{n} \ldots \dot{x}_{j+1} \dot{x}_{j-1} \ldots \dot{x}_{1}
$$

Thus, for each term of an expansion which has a representation of the form $<i_{n} \ldots i_{j+1} 0 i_{j-1} \ldots i_{1}>$ then a new term is generated which is represented by the $n$-tuple $\left\langle i_{n} \ldots i_{j+1} 1 i_{j-1} \ldots i_{1}\right\rangle$. The newly generated terms are then compared with the existing terms. If any terms are found to be equivalent then both
the newly generated term and the term existing in the expansion are deleted. Any generated term which has no equivalent term should be added to the representation of the expansion. The procedure is repeated, generating new terms of the form $\left\langle i_{n} \ldots i_{k+1} 1 i_{k-1} \ldots i_{1}\right\rangle(k \in\{1,2, \ldots, n\} k \neq j)$ and deleting equivalent terms. The conversion is complete when this procedure has been applied for each function variable. The final representation is the RM expansion. The transformation from a Boolean function to a RM expansion is comprised of $n$ steps.

The number of steps in transforming a RM expansion to a FPRM expansion is dependent on the polarity of the FPRM expansion. Indeed, it is equal to the total number of 1 's in the binary representation of the polarity number $p,\left\langle p_{n} p_{n-1} \ldots p_{1}\right\rangle$. If $p_{j}=1(j \in\{1,2, \ldots, n\})$ then the variable $x_{j}$ must be present in complemented form throughout the FPRM expansion. The equality

$$
x_{j}=\bar{x}_{j} \oplus 1
$$

may be expressed in the form

$$
\dot{x}_{n} \ldots x_{j} \ldots \dot{x}_{1}=\dot{x}_{n} \ldots \bar{x}_{j} \ldots \dot{x}_{1} \oplus \dot{x}_{n} \ldots \dot{x}_{j+1} \dot{x}_{j-1} \ldots \dot{x}_{1}
$$

Thus, if $p_{j}=1$ and any term of a RM expansion has a representation of the form $\left\langle i_{n} \ldots i_{j+1} 1 i_{j-1} \ldots i_{1}\right\rangle$ then generate a new term which is represented by the $n$-tuple $\left\langle i_{n} \ldots i_{j+1} 0 i_{j-1} \ldots i_{1}\right\rangle$. The procedure of deleting equivalent terms and adding any remaining generated terms to the existing expansion is identical to that described for the process of transforming a Boolean function to a RM expansion. The conversion process is complete when the procedure has been applied for each variable $x_{j}$ for which $p_{j}=1$.

The tabular techniques presented in [1] include a heuristic algorithm which derives (minimal) sub-minimal FPRM expansions of Boolean functions. The minimisation procedure commences by evaluating the RM expansion (polarity 0 FPRM expansion) and proceeds by complementing expansion variables in order to reduce the number of product terms in the expansion. The final representation is a FPRM expansion. The algorithm comprises of a series of distinct steps.
S. 1 Count the number of occurrences of each variable $x_{j}$ (this is
equivalent to counting the number of 1 's in each column of the tabular representation of the FPRM expansion). Let this equal occur_ $x_{f}$ This value, occur_ $x_{f}$, equals the number of new product terms generated if the state of variable $x_{j}$ is altered ( $x_{j}-\bar{x}_{j}$ ).
S. 2 Determine the number of product terms which may be deleted (i.e. equivalent product terms) by complementing a variable. This is realised by counting the number of pairs of product terms which are adjacent in each variable $x_{f}$, and is denoted adj_ $x_{j}$. (Two product terms $\rho_{g}$ and $\rho_{b}$ are said to be adjacent in variable $x_{j}$ iff $\rho_{a}=\dot{x}_{n} \ldots \dot{x}_{j} \ldots \dot{x}_{1}, \rho_{b}=\dot{x}_{n} \ldots \dot{x}_{j+1} \dot{x}_{j-1} \ldots \dot{x}_{1}$ and $\rho_{a}=\dot{x}_{j} \rho_{b}$ Hence, $\rho_{a} \oplus \rho_{b}=\dot{x}_{j} \rho_{b} \oplus \rho_{b}=\rho_{b}\left(\dot{x}_{j} \oplus 1\right)\left(a, b \in\left\{0,1, \ldots, 2^{n}-1\right\}\right.$, $j \in\{1,2, \ldots, n\}$ ).
S. 3 Calculate diff $_{-} x_{j}$, the number of product terms which can be lost from or gained by the FPRM expansion by complementing variable $x_{j}$,

$$
d i f f_{-} x_{j}=o c c u r_{-} x_{j}-\left(2 \times a d j_{-} x_{j}\right)
$$

S. 4 Find variable $x_{j}$ for which diff $x_{j}$ is a minimum. Let this equal min_diff.
If min_diff < 0 then $\mid$ min_diff $\mid$ product terms will be lost from the FPRM expansion by complementing variable $x_{j}$ for which diff $x_{j}=$ min_diff. Hence, convert the FPRM expansion to the new FPRM expansion with variable $x_{j}$ complemented.
If min_diff $\geq 0$ then the number of product terms in the FPRM expansion is minimal (sub-minimal) and cannot be reduced by complementing any single variable $x_{j}$
This series of steps can be applied to the new FPRM expansion to determine whether the number of product terms in the expansion can be further reduced. This may be repeated until the number of product terms in the expansion cannot be reduced.

The tabular technique for determining minimal (sub-minimal) FPRM expansions is similar to an earlier method developed by Marinkovic and Tosic [21]. The algorithms which they proposed employed the technique subsequently adopted by Almaini et al [1]. That is, determining which expansion variables should be complemented by evaluating the number of occurrences of variables and the number of pairs of adjacent product terms. Two algorithms are presented, one is heuristic whilst the second determines the minimum FPRM expansion without performing an exhaustive
search. The first algorithm is very similar to the tabular technique [1] reviewed previously. The tabular technique minimisation method does not allow an expansion variable which has been complemented during the minimisation procedure to be returned to its true state. Indeed, once an expansion variable has been complemented it is excluded from the evaluation process. Although not explicitly stated in the algorithm presented in [21], it was implied that expansion variables may be returned to their true state if this conversion further reduces the number of terms in a FPRM expansion. The second algorithm operates in a manner similar to the first. However, the FPRM expansions determined as minimal are always the absolute minimum and not sub-minimal expansions. The solutions may be found without performing exhaustive searches but this cannot be guaranteed. They are derived through eliminating FPRM expansions which are not minimum forms from the evaluation procedure.

The tabular technique minimisation procedure and algorithms A1 and A2 presented in [21] are now briefly summarised. It may be possible to merge the strengths of each algorithm to form an improved heuristic minimisation technique.

## Summary of tabular technique minimisation algorithm

1.a The maximum possible number of iterations of the algorithm is equal to the number of expansion variables.
1.b If several expansion variables, $x_{j}$, have diff_ $x_{j}=$ min_diff then the choice of the variable to be complemented is arbitrary. This choice obviously affects the quality of the final solution.
1.c During evaluation, if min_diff $=0$ then the algorithm will cease. If, however, the state of variable $x_{j}\left(d i f f_{-} x_{j}=0\right)$ is altered then a new FPRM expansion will be generated. Although the number of product terms in the representation has not been reduced another iteration of the algorithm can be performed introducing the possibility of further reducing the number of product terms in the expansion.
1.d Expansion variables may only be complemented and cannot be returned to their true state. It is possible that allowing a variable which has been complemented during an earlier iteration of the minimisation algorithm to be returned to its true state may enhance the quality of the final solution.
1.e The starting point of the minimisation algorithm is the RM expansion (polarity 0 FPRM expansion). If the minimal FPRM expansion contains many complemented variables then many iterations of the algorithm must be performed increasing both the possibility of locating a locally minimum FPRM expansion rather than the global minimum and the time taken to reach a solution.
1.f The algorithm will locate and cease on locally minimum FPRM expansions and cannot determine if a global minimum exists.

## Summary of algorithm A1

This algorithm is structurally very similar to the tabular technique minimisation algorithm, however, its operation is ambiguous in certain areas.
2.a If, for more than one expansion variable, altering the state of each variable maximally reduces the number of product terms in a FPRM expansion then the algorithm does not explicitly state which variable should be altered. (See 1.b.)
2.b If altering the state of any variable of a FPRM expansion will not reduce the number of product terms in the FPRM expansion then the algorithm will cease. (See 1.c.)
2.c It is unclear whether an expansion variable complemented during an earlier iteration of the minimisation algorithm may be returned to its true state. (See 1.d.)
2.d The FPRM expansion with which the minimisation algorithm commences is determined by the user. (See 1.e.)
2.e The algorithm will locate and cease on locally minimum FPRM expansion and cannot determine if a global minimum exists. (See 1.f.)

## Summary of algorithm A2

This algorithm will always determine the minimal FPRM expansion. The technique evaluates the number of product terms in all FPRM expansions without it being necessary to generate each FPRM expansion.

The following modifications are derived from observing the strengths and weaknesses of the tabular technique minimisation algorithm and algorithms A1 and A2 summarised previously.

## Modifications

a Introduction of $a$ branching mechanism into the algorithm. If $k$ variables have diff_ $x_{j}=$ min_diff then generate $k$ new FPRM expansions. Each of the $k$ new FPRM expansions is of a different polarity and in each the state of a single expansion variable differs from its state in the original FPRM expansion. During the next iteration of the minimisation algorithm the FPRM expansions which can be maximally reduced are used to generate new FPRM expansions. Any remaining FPRM expansions are deleted. (See 1.b, 2.a.)
b If diff_ $x_{j}=$ min_diff $=0$ then the state of expansion variable $x_{j}$ is altered. This does not reduce the number of product terms in the solution but generates new FPRM expansions allowing further iterations of the algorithm. (See 1.c, 2.b.)
c The state of an expansion variable may be altered more than once. (See 1.d, 2.c.)
These modifications were applied to the tabular technique minimisation algorithm to realise the Full Gains minimisation algorithm. This new technique and results illustrating its performance are presented in the following sections.

### 3.2.2 Full Gains Minimisation Algorithm

The definitions of the following terms are based on those of the tabular technique minimisation algorithm described in section 3.2.1.

Let occur_x $x_{p, j}$ denote the number of occurrences of expansion variable $x_{j}$ in the polarity $p$ FPRM expansion. ( $p \in\left\{0,1, \ldots, 2^{n}-1\right\}, j=1,2, \ldots, n$ )

Term adj_x ${ }_{p, j}$ denotes the number of pairs of terms in the polarity $p$ FPRM expansion which are adjacent in variable $x_{j}$
The term diff_ $x_{p, j}$ represents the number of terms which may be lost from or gained by the polarity $p$ FPRM expansion by altering the state of variable $x_{j}$ Hence, $\operatorname{diff}_{-} x_{p, j}=$ occur_ $_{p_{p, j}}-\left(2 \times \operatorname{adj} x_{p, j}\right)$
If diff $_{-} x_{p, j}<0$ then $\mid$ diff $_{-} x_{p, j} \mid$ product terms will be lost from the polarity $p$ FPRM expansion by altering the state of variable $x_{j}$ If diff_x $x_{p, j}>0$ then $\mid$ diff_ $_{-} x_{p, j} \mid$ product terms will be gained by the polarity $p$ FPRM expansion by altering the state of variable $x_{f}$ If diff_ $_{x_{p, j}}=0$ then altering the state of variable $x_{j}$ will have no effect on the number of product terms in the polarity $p$ FPRM expansion.
$P$ is the set of FPRM expansions which are minimal (sub-minimal), $P \leq$
$\left\{0,1, \ldots, 2^{n}-1\right\}$.
$R$ is the set of FPRM expansions which are minimal (sub-minimal), $R ⿷$ $\left\{0,1, \ldots, 2^{n}-1\right\}$.
$Q$ is the set of all FPRM expansions generated by the minimisation algorithm, $Q \subseteq\left\{0,1, \ldots, 2^{n}-1\right\}$.
The minimum value of diff_ $_{p, j}$ is min_diff where $p \in \mathrm{R}, j=1,2, \ldots, n$.

## Full Gains minimisation algorithm

S. $1 \quad$ Set $P=\{ \}$ and $R=\{ \}$.
S. 2 Convert the Boolean function to the equivalent RM expansion (Polarity $p=0)$. Set $P=\{0\}, R=\{0\}, Q=\{0\}$.
S. 3 Determine occur_x $x_{p, j}$ and adj$x_{p, j}$ for all $p \in R$ and $j=1,2, \ldots n$.
S. 4 Calculate diff_x $x_{p, j}$ for all $p \in R$ and $j=1,2, \ldots n$.
S. 5 Find min_diff. If min_diff > 0 then go to S. 9 else go to S.6.
S. 6 Find all FPRM expansions with $d_{\text {iff_ }} x_{p, j}=\min _{-} d i f f, p \in R$ and $j=$ 1,2,...,n.
S. 7 If min_diff $<0$ then set $P=\{ \}$ and insert into $P$ the polarity numbers, $p$, of each FPRM expansion with diff_x $x_{p, j}=$ min_diff, as found in S.6, removing any duplicates. Repeat for set $R$.

If min_diff $=0$ then add to set $P$ any polarity numbers, $p$, of the new FPRM expansion generated in $\mathbf{S .} 6$ which are not currently contained in $P$. Set $R=\{ \}$, if any polarity numbers, $p$, of the new FPRM expansions have been added to $P$ then insert these numbers into $R$. Otherwise $R=\{ \}$.
Insert into $Q$ the polarity numbers, $p$, of each FPRM expansion with diff $_{-} x_{p, j}=$ min_diff, as found in S.6, removing any duplicates. $^{\text {a }}$ If min_diff > 0 then go to $\mathbf{S} .9$
S. 8 If $Q \neq\left\{0,1, \ldots, 2^{n}-1\right\}$ and $R \neq\{ \}$ then for each FPRM expansion with diff_$x_{p, j}=$ min_diff generate new FPRM expansions in which the state of variable $x_{j}$ is altered ( $\dot{x}_{j}-\bar{x}_{j}$ ). Go to S.3. Otherwise go to S.9.
S. 9 The algorithm determines the polarity $p$ FPRM expansions to be minimal, $p \in P$.

It is suggested that if this new algorithm is employed in preference to either the tabular technique minimisation algorithm or the first algorithm presented by Marinkovic and Tosic then it is more probable that minimal

FPRM expansions will be derived. However, the technique requires more computation time as it introduces the possibility of performing more iterations of the algorithm and operating on more that one FPRM expansion during each iteration. Also, the number of FPRM expansions generated by the algorithm cannot be controlled depending instead on the inherent structure of the Boolean function. These considerations led to further refinement of the new algorithm with the goal of increasing the efficiency of the technique. The new algorithm was modified by removing the facility to transform a polarity $p$ FPRM expansion to new FPRM expansions if min_diff $=0$. This is realised by replacing $S .5$ and S. 7 of the algorithm with two new steps.
S.5' Find min_diff. If min_diff $\geq 0$ then go to S .9 else go to S.6.
S.7' If min_diff $<0$ then set $P=\{ \}$ and insert into $P$ the polarity numbers, $p$, of each FPRM expansion with diff_x $x_{p, j}=$ min_diff, as found in S.6, removing any duplicates. Repeat for set $R$. Insert into $Q$ the polarity numbers, $p$, of each FPRM expansion with diff_ $_{p, j}=$ min_diff, as found in S.6, removing any duplicates. $_{\text {, }}$ If min_diff $\geq 0$ then go to S. 9

The branching mechanism and the facility to repeatedly alter the state of each variable remain unchanged. The modified form of the Full Gains minimisation algorithm is entitled the Full Gains Min0 minimisation algorithm.

The graphs of Figure 3.1 - Figure 3.3 illustrate the performance of four different minimisation algorithms. Two of the algorithms evaluated are the Full Gains and Full Gains Min0 minimisation algorithms described previously. The Tabular technique is the minimisation algorithm reviewed in section 3.2.1. The Boolean matrix optimisation method was developed by Habib [41] and is a heuristic technique which determines minimal (sub-minimal) FPRM expansions of Boolean functions. The graphs display as a percentage the number of Boolean functions for which each optimisation algorithm derived minimal FPRM expansions. The minimal FPRM expansions of each Boolean function were determined using the technique developed by Harking [42]. The $x$-axis of each graph illustrates the number of variables and minterms in a Boolean function. The results presented in Figure 3.1 and Figure 3.3 are derived from testing each algorithm with sets of 1000 randomly
generated Boolean functions with fixed numbers of variables and minterms. The Boolean functions were constructed by a random number generator. The output of this random number generator was filtered so as to remove any duplicate minterms. In Figure 3.2, each algorithm optimised sets comprised of all possible Boolean functions which could be constructed from the indicated numbers of variables and minterms. All algorithms (a. Full Gains b. Full Gains Min0 c. Boolean matrix d. Tabular technique e. exhaustive search (Harkings' technique [42]) were implemented in Pascal and the programs executed on a HP workstation.

It may be observed from each of the graphs in Figure 3.1 - Figure 3.3 that the Full Gains minimisation algorithm consistently produces superior results. This is a predictable outcome due to the structure of this algorithm and that of the two others which were tested, namely Tabular technique and Full Gains Min0. The performance of the Boolean matrix minimisation algorithm degrades very rapidly as the numbers of variables in the Boolean functions increases. It produced poor results when compared with the other algorithms.

The effectiveness of the branching mechanism (modification a) is to some extent illustrated by the improved performance of Full Gains Min0 when compared with the performance of Tabular technique. Consider the modification allowing the state of variables to be altered without causing a reduction in the number of product terms in the FPRM expansion (modification b). The effects of this modification are illustrated by the differences in performance between Full Gains and Full Gains Min0. Unfortunately, from these graphs it is impossible to determine the effects of the modification allowing the repeated alterations in the state of expansion variables (modification $c$ ). These results illustrate that two out of three modifications introduced to the original tabular technique minimisation algorithm have proved to be effective. It is suggested that modification $c$ has also acted to improve the quality of the solutions determined by minimisation algorithms Full Gains and Full Gains Min0.

The following table (Table 3.1) presents results which illustrate the performance of the Full Gains minimisation algorithm when operating on a number of different Boolean functions. These functions are selected from


Figure 3.1: Percentage of randomly generated Boolean functions for which the minimisation algorithms formed optimum FPRM expansions. (Boolean functions of 4 variables, $3-13$ minterms ( 1000 Boolean functions per set).)

Figure 3.2: Percentage of Boolean functions for which the minimisation algorithms formed optimum FPRM expansions. (All Boolean functions of 4 variables, 3-13 minterms.)

|  |  | $\begin{aligned} & \text { 旨 } \\ & \stackrel{0}{3} \\ & \overline{3} \end{aligned}$ |  |
| :---: | :---: | :---: | :---: |
|  |  |  |  |



Figure 3.3: Percentage of randomly generated Boolean functions for which the minimisation algorithms formed optimum FPRM expansions. (Boolean functions of 4-10 variables.)
the 1991 set of benchmarks distributed by the MCNC ${ }^{\dagger}$. Generally, each function referred to in the table has been adapted from a multiple-output function (as circulated by the MCNC) to a single-output function suitable for use with the minimisation algorithms listed in the table.

|  |  | io. of prod- | No. od product terss in minimal (sub-sininal) PPRM expansion |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Bookean function | No. ${ }^{2}$ variables $\square$ | Espresso-II [63] | Exhaustive search [42] | Tabular <br> technique <br> [1] | Bookean <br> matrix <br> [41] | Pull Cains | CCRMI <br> [46] |
| 5xpll | 7 | 7 | 12 | 12 | 16 | 12 | 12 |
| 9syn | 9 | 85 | 173 | 182 | 173 | 182 | 173 |
| bw7 | 5 | 6 | 8 | 12 | 14 | 12 | 12 |
| conl2 | 7 | 5 | 8 | 8 | 9 | 8 | 8 |
| $51 . \mathrm{m} 4$ | 8 | 10 | 7 | 7 | 9 | 7 | 9 |
| rd332 | 5 | 16 | 5 | 5 | 5 | 5 | 5 |
| rd732 | 5 | 64 | 7 | 7 | 7 | 7 | 7 |
| r8842 | 8 | 128 | 8 | 8 | 8 | 8 | 8 |
| S0022 | 10 | 20 | 52 | 52 | 52 | 52 | 61 |
| S0023 | 10 | 22 | 47 | 47 | 61 | 47 | 59 |
| 242] | 7 | 28 | 9 | 9 | 9 | 9 | 13 |

Table 3.1
The first and second columns of this table indicate the title of the Boolean functions and the number of variable in the functions, respectively. The results presented in the third column indicate the number of products in the SOP expansions subsequent to minimisation using Espresso-II [63]. Espresso-II derives a minimal (sub-minimal) SOP representation of an initial Boolean function. The benchmark functions were also optimised using four other heuristic minimisation algorithms, hence the values presented in the relevant columns of the Table 3.1 indicate the number of product terms in the minimal (sub-minimal) FPRM expansions. The CGRMIN algorithm was devised by Sarabi and Perkowski [46] and is discussed in section 3.1. This heuristic technique determines minimal (sub-minimal) FPRM expansions of Boolean functions. Finally, the number of product terms in the absolute minimum FPRM expansion was determined by exhaustive search and is
presented in the fourth column of Table 3.1.

The results presented in Table 3.1 indicate that the Full Gains minimisation algorithm performs satisfactory when compared with other minimisation techniques. In general, it performs better than that of the Boolean matrix and CGRMIN minimisation techniques. It is, however, interesting to note that for this small set of benchmark functions both the Tabular technique and Full Gains minimisation algorithms produced FPRM expansions comprised of equivalent numbers of product terms. It is also interesting to compare the numbers of product terms in the Boolean SOP forms determined by Espresso and the numbers of product terms in the corresponding FPRM expansions.
3.2.3 Pre-treatment Techniques for the Full Gains Minimisation Algorithm When reviewing the tabular technique minimisation algorithm it was noted that the algorithm commenced by evaluating the polarity 0 FPRM expansion. If the polarity, p , of the optimum FPRM expansion has many bits $p_{j}=1$ then the algorithm will cease at any local minimum which exists between the polarity 0 FPRM expansion and the optimum polarity $p$ FPRM expansion. Additionally, many iterations of the algorithm must be performed before the solution is reached. In order to overcome these problems two pre-treatment techniques have been devised and implemented. These pre-treatment techniques determine the polarity $p$ of the FPRM expansion to which the initial Boolean function should be transformed and hence, the FPRM expansion initially evaluated by the minimisation algorithm.

The pre-treatment techniques may also be used in conjunction with the algorithm A1 [21] as it was noted that the polarity $p$ of the initial FPRM expansion is undefined. The FPRM expansion obtained by a pre-treatment technique may be used as the starting FPRM expansion.
The pre-treatment techniques are now introduced.

## Pre-treatment

If, for a Boolean function of $n$ variables, the total number of variables present in true form is less than the number of variables present in complemented form then convert the Boolean function to the negative polarity (polarity $2^{n}-1$ ) FPRM expansion. Otherwise,
convert the Boolean function to the positive polarity (polarity 0 ) FPRM expansion. This FPRM expansion is the initial expression evaluated by the minimisation algorithm.
The pre-treatment technique evolved from observing the process of converting a Boolean function to a FPRM expansion. It was noted that if the majority of variables in a Boolean function are present in true form then converting the function to the polarity 0 FPRM expansion may generate fewer terms than converting the Boolean function to the polarity $2^{n}-1$ FPRM expansion. The pre-treatment technique does not take into account the number of minterms lost through deleting equivalent terms but concentrates solely on minimising the increase in the size of the FPRM expansion through limiting the generation of product terms.

This pre-treatment technique was tested as a supplement to the Full Gains minimisation algorithm (S.1 of the algorithm being modified). In general, the results of the tests indicated that employing the Full Gains minimisation algorithm with pre-treatment led to an increase in the number of occasions on which the algorithm derived the minimal FPRM expansion. However, it was noted that the pre-treatment was unsuccessful for Boolean functions which comprised of larger numbers of minterms, relative to the number of function variables. The initial pre-treatment technique was based on limiting the number of terms generated during the conversion from Boolean function to FPRM expansion. Whilst this approach seemed suitable for functions with low numbers of minterms (relative to the number of function variables) it was conjectured that this may not be valid for functions with larger numbers of minterms. Through observation, it was determined that a more appropriate technique seemed to be that of reducing the number of terms in the final FPRM expansion through deleting equivalent terms during the conversion process. Hence, a Boolean function with a large number of minterms should be converted to the polarity $0\left(2^{n}-1\right)$ FPRM expansion if the majority of variables are present in complemented (true) form. This would increase the numbers of new product terms generated thus increasing the possibility of deleting equivalent product terms. This led to a modified form of pre-treatment which is now described.

## 1) Pre-treatment Pos_neg

If the number of minterms in the $n$ variable Boolean function is less than or equal to half the total possible number of minterms which
may be formed (i.e. $2^{n}$ ) then 1.a, else 1.b.
1.a) If the total number of variables present in the Boolean function in true form is less than the number of variables present in complemented form then convert the Boolean function to the negative polarity (polarity $2^{n}-1$ ) FPRM expansion. Otherwise convert the Boolean function to the positive polarity (polarity 0) FPRM expansion.

This FPRM expansion is the initial expression evaluated by the minimisation algorithm.
1.b) If the total number of variables present in the Boolean function in true form is less than the number of variables present in complemented form then convert the Boolean function to the polarity 0 FPRM expansion (PPRM expansion). Otherwise convert the Boolean function to the negative polarity (polarity $2^{n}-1$ ) FPRM expansion.
This FPRM expansion is the initial expression evaluated by the minimisation algorithm.

As a further modification a pre-treatment technique was devised whereby the resulting FPRM expansion could be of any polarity $p$ ( $p \in\left\{0,1, \ldots, 2^{n}-1\right\}$ ). The technique entails evaluating the state of each function variable.
2) Pre-treatment M_pre_pol

If the number of minterms in the $n$ variable Boolean function is less than or equal to half the total possible number of minterms which may be formed (i.e. $2^{n}$ ) then 2.a, else 2.b.
2.a) For each function variable $x_{j}(j=1,2, \ldots, n)$,
if, for the Boolean function, the number of occurrences literal $x_{j}$ (true form) is less than the number of occurrences of literal $\bar{x}_{j}$ (complemented form) then $p_{j}=$ 1. Otherwise $p_{j}=0$.
2.b) For each function variable $x_{j}(j=1,2, \ldots, n)$,
if, for the Boolean function, the number of occurrences
literal $x_{j}$ (true form) is less than the number of occurrences of literal $\bar{x}_{j}$ (complemented form) then $p_{j}=$ 0 . Otherwise $p_{j}=1$.
Form the polarity number $p$, the decimal equivalent of $\left\langle p_{n} p_{n-1} \ldots p_{1}\right\rangle$. Convert the Boolean function to the polarity $p$ FPRM expansion. This

FPRM expansion is the initial expression evaluated by the minimisation algorithm.

The pre-treatment techniques, Pre-treatment Pos_neg and Pre-treatment M_pre_pol, were tested as supplements to the Full Gains minimisation algorithm. An additional pre-treatment technique identified as Pre-treatment Pre_pol was also tested. This procedure is similar to M_pre_pol, however, the number of minterms in the Boolean function has no effect on the polarity of the Initial FPRM expansion. Thus only step 2.a of the Pretreatment M_pre_pol is utilised. The procedure for testing the pretreatment techniques is now outlined.

Perform pre-treatment (Pos_neg, M_pre_pol or Pre_pol) to determine the polarity $p$ of the initial FPRM expansion.
S. 1 as for Full Gains minimisation algorithm.
S.2' Set $P=\{p\}, R=\{p\}, Q=\{p\}$ where $p$ is determined by the pre-treatment and is the polarity of the FPRM expansion used in the first iteration of this algorithm.
S. 3 - S. 9 as for Full Gains minimisation algorithm.

The graphs of Figure 3.4 and Figure 3.5 illustrate the performance of the three pre-treatment techniques when used in conjunction with the Full Gains minimisation algorithm. Results generated by the Full Gains minimisation algorithm without pre-treatment are also presented. The graphs display as a percentage the number of Boolean functions for which each minimisation technique derived minimal FPRM expansions. The minimal FPRM expansions of each Boolean function were determined using the technique developed by Harking [42]. The x-axis of each graph illustrates the number of variables and minterms in a Boolean function. The results are derived from testing each algorithm with sets of 1000 randomly generated Boolean functions with fixed numbers of variables and minterms. As for the previous sets of result the Boolean functions were constructed by a random number generator. The output of the random number generator was filtered so as to remove any duplicate minterms. The algorithms (a. Full Gains b. Pos_neg Full Gains c. Pre_pol Full Gains d. M_pre_pol Full Gains e. exhaustive search - Harkings' technique [42]) were implemented in Pascal and the programs executed on a HP workstation.

The graphs presented in Figure 3.4 and Figure 3.5 illustrate the benefits of introducing a pre-treatment step into the Full Gains minimisation algorithm. In general, both pre-treatments Pos_neg and M_pre_pol improve the performance of the Full Gains minimisation algorithm. The pre-treatment technique Pre-pol degrades the performance of the Full Gains minimisation algorithm when used to minimise Boolean functions with large numbers of minterms (relative to the total number of minterms which may be used to represent any Boolean function, i.e. $2^{n}$ ). This result would seem to support the previous observations on which the structure of the pre-treatment techniques M_pre_pol and Pos_neg are based.

### 3.3 Summary

The heuristic minimisation algorithms presented in this chapter may be employed to determine optimal (sub-optimal) FPRM expansions of Boolean functions. The first technique described, the Full Gains minimisation algorithm, is an extension to an existing technique developed by Marinkovic and Tosic [21]. Additionally, the notation used to represent both Boolean functions and FPRM expansions and the technique used to generated FPRM expansions are those developed by Almaini et al [1]. A modified form of the Full Gains minimisation algorithm has been presented, namely the Full Gains Min0 minimisation algorithm. This technique may be considered to be a reduced form of the initial algorithm as certain steps have been omitted. This will to some extent limit both the number of FPRM expansions which will be generated during the minimisation operation and the number of iterations of the algorithm. Both the Full Gains minimisation algorithm and its modified form have undergone extensive evaluation and results have been presented illustrating the performance of these algorithms when compared with existing heuristic techniques. The significance of these results has been explored in section 3.2.2.

Pre-treatment techniques suitable for use with the Full Gains and Full Gains Min0 minimisation algorithms have also been devised. The purpose of the pre-treatment is to provide a means of predicting a sub-optimal FPRM expansion of a Boolean function which would be used as a starting expansion by either the Full Gains or Full Gains Min0 minimisation algorithms. Results are presented which illustrate the performance of the Full Gains minimisation algorithm when used in conjunction with the pre-


Figure 3.4: Percentage of randomly generated Boolean functions for which the minimisation techniques (pre-treated) formed optimum FPRM expansions. (Boolean functions of 4 variables, 3-13 minterms (1000 Boolean functions per set).)


Figure 3.5: Percentage of randomly generated Boolean functions for which the minimisation techniques (pre-treated) formed optimum FPRM expansions. (Boolean functions of 4-10 variables.)
treatment techniques. An evaluation of these results is given in section 3.2.3.

The heuristic algorithms and techniques introduced in this chapter form alternative means of deriving optimal (sub-optimal) FPRM expansions of completely specified Boolean functions. In general, these methods show improved performance when compared with a limited number of existing techniques. However, it is possible that the new techniques are more complex than some established methods.

## Chapter 4

## Fixed Polarity Reed-Muller Expansions of Incompletely Specified Boolean Functions'

The generation of FPRM expansions from incompletely specified Boolean functions is discussed in this chapter. Initially, the concept of the ReedMuller 'don't care' term is explored and existing techniques for generating and allocating RM 'don't care' terms are reviewed. A technique which finds the optimum allocation of the 'don't care' terms leading to minimal FPRM expansions is then described. The problem of determining the optimum allocation of 'don't care' terms whilst addressing the problem of deriving the optimal polarity is also addressed. Two heuristic approaches to solving this problem are presented.

### 4.1 Fixed Polarity Reed-Muller Expansions of Incompletely Specified Boolean Functions

An incompletely specified Boolean function has one or more input conditions for which the corresponding output states are undefined. Any incompletely specified $n$ variable Boolean function may be represented as

$$
f\left(x_{n}, x_{n-1} \cdots, \ldots, x_{1}\right)=d_{0} \bar{x}_{n} \bar{x}_{n-1} \ldots \bar{x}_{2} \bar{x}_{1}+d_{1} \bar{x}_{n} \bar{x}_{n-1} \ldots \bar{x}_{2} x_{1}+d_{2} \bar{x}_{n} \bar{x}_{n-1} \ldots x_{2} \bar{x}_{1}+\ldots \ldots+d_{2 n-1} x_{n} x_{n-1} \ldots x_{2} x_{1}
$$

$d_{i} \in\{0,1, \mathrm{D}\}$ is an operational domain coefficient which may take the value 0,1 or the undefined state, $D(D \in\{0,1\})$.
$x_{j}$ and $\bar{x}_{j}$ are literals of the function, in true and complemented forms respectively.
$i=0,1, \ldots, 2^{n}-1, j=1,2, \ldots, n$
If the value of a coefficient, $d_{i}$, is defined (it is either 0 or 1 ) then the minterm, $m_{i}$, associated with the coefficient is a specified term of the function. If, however, the value of $d_{i}$ is undefined, that is, it may take the

[^1]value 0 or 1 without affecting the output of the function, then the corresponding minterm, $m_{i}$, is an unspecified or 'don't care' term of the function. The 'don't care' terms offer a degree of freedom when minimising Bóolean SOP forms.

The RM expansion of a Boolean function may be derived from the Boolean SOP form (chapter 2). The RM (PPRM) expansion of $n$ variables has the form

$$
f\left(x_{n}, x_{n-1} \ldots, x_{1}\right)=a_{0} \oplus a_{1} x_{1} \oplus a_{2} x_{2} \oplus a_{3} x_{2} x_{1} \oplus \ldots \ldots \odot a_{2-1} x_{n} x_{n-1} \ldots x_{2} x_{1}
$$

The functional domain coefficients, $a_{i}$, are related to the operational domain coefficients, $d_{i}$, in the following manner,

$$
\begin{aligned}
& a_{0}=d_{0} \\
& a_{1}=d_{0} \oplus d_{1} \\
& a_{2}=d_{0} \oplus d_{2} \\
& a_{3}=d_{0} \oplus d_{1} \oplus d_{2} \oplus d_{3} \\
& \cdot \\
& \cdot \\
& a_{2^{n}-2}= \\
& a_{2^{n}-1}=d_{0} \oplus d_{0} \oplus d_{1} \oplus d_{4} \oplus d_{2} \oplus d_{6} \oplus \ldots \ldots \oplus d_{3} \oplus \ldots \ldots \oplus d_{2^{n}-4} \oplus d_{2^{n_{-2}}} \\
& \cdot
\end{aligned}
$$

If a coefficient $d_{i}$ is undefined then any coefficient $a_{k}\left(i, k \in\left\{0,1, \ldots, 2^{n}-1\right\}\right)$ which is dependent on $d_{i}$ will also be undefined. The piterm associated with an undefined coefficient $a_{k}$ may be described as an unspecified piterm and the RM expansion may be denoted an incompletely specified RM expansion. As Boolean 'don't care' terms may be utilised to form minimal SOP expansions, it follows that RM 'don't care' terms may also be employed to minimise RM expansions. The minimum form of a RM expansion is that which has the minimum weight, that is, the least number of piterms. Hence, the goal in minimising an incompletely specified RM expansion is to set the maximum number of coefficients, $a_{k}$, equal to 0 . As these coefficients are dependent on the coefficients $d_{1}$, the minimisation problem may be formulated as deriving the assignment of Boolean 'don't care' terms which results in the maximum number of coefficients $a_{k}$ being equated to 0 . The relationship between the coefficients $d_{i}$ and $a_{k}$ is not a one-to-one mapping. Thus, it may be conjectured that it is not generally the case that the allocation of 'don't care' terms which leads to a minimum Boolean SOP form
will also lead to a minimum RM expansion [37].

The particular case of the polarity 0 FPRM expansion described previously may be extended to the other $2^{n}-1$ FPRM expressions. The coefficients of each FPRM expansion, $b_{k^{\prime}}$, can be related to the coefficients, $d_{i}$, of the Boolean SOP form as discussed in chapter 2. Therefore, an incompletely specified Boolean function can be transformed into an equivalent incompletely specified FPRM expansion. The minimal FPRM expansion is that which has the maximum number of coefficients $b_{k}$ equal to 0 . The techniques and algorithm developed in a later section of this chapter may be employed to determine minimal FPRM expansions of incompletely specified Boolean functions. Note, the polarity of each FPRM expansion is predetermined.

It is, perhaps, necessary to state that the unspecified terms of a Boolean function may be used to minimise ESOP forms. This is best explained by considering the Karnaugh representation of a Boolean function where each cell of the map represents a minterm of the function. Obviously, an ESOP form may be represented using a map where the relationship between the product terms represented by each cell of the map is modulo-2 addition. On a Karnaugh map representing a Boolean function, specified and unspecified terms may be grouped according to the rules of Boolean algebra to obtain a minimised SOP form of the original function. A similar technique may be applied to the map representing the ESOP expression, however, specified and unspecified terms must be grouped according to the algebra of $\mathrm{GF}(2)$. That is, any cell containing the value 1 may be included in an odd number of groups, whilst any cell containing the value 0 may be excluded from all groups or included in an even number of groups. A cell which contains the value $D$, representing a 'don't care' term, may be included in any number of groups. An example of the technique is given in Figure 4.1. Determining the allocation of the 'don't care' terms so as to derive a minimal ESOP form is not a trivial task and continues to be a research topic.

### 4.2 Review of Techniques for Deriving Fixed Polarity Reed-Muller Expansions of Incompletely Specified Boolean Functions <br> The literature review of the previous chapter (section 3.1) examined

$$
f\left(x_{3}, x_{2}, x_{1}\right)=\bar{x}_{3} \bar{x}_{2} x_{1}+\bar{x}_{3} x_{2} \bar{x}_{1}+x_{3} \bar{x}_{2} x_{1}+x_{3} x_{2} x_{1}+D \bar{x}_{3} \bar{x}_{2} \bar{x}_{1}
$$

Karnaugh map representation of Boolean function


Groupings using rules of Boolean algebra.

Map representation of equivalent ESOP expansion.


Groupings using rules of GF(2) algebra.


Figure 4.1: Minimisation of a Boolean function and an ESOP expression using 'don't care' terms.
techniques for representing, generating and optimising FPRM expansions. Many of the methods discussed included techniques suitable for incompletely specified Boolean functions, these techniques are now described in more detail.

Tran [36, 38] described a method for transforming an incompletely specified Boolean function represented on a Karnaugh map to a FPRM expansion. The method is an extension to the map folding technique and the resulting FPRM expansion is represented on a RM-coefficient map. The 'don't care' terms of the Boolean function are transformed to RM 'don't care' terms where any RM 'don't care' term may be dependent on one or more of the unspecified terms of the initial Boolean function. These RM 'don't care' terms may then be allocated values of 0 or 1 so as to minimise the number of cells of the map which contain the value 1 . Tran did not suggest an algorithm for this purpose and user must decide on the best values for the 'don't care' terms in order to derive the FPRM expansion comprised of the minimum number of product terms. This map method becomes impractical as the number of variables in the Boolean function increases.

The RM coefficient map and map folding was utilised by Green [37] who illustrated a technique for transforming an incompletely specified Boolean function to a FPRM expansion. The Boolean function is partitioned into two subfunctions, where one subfunction comprises of the specified minterms of the original function. The second subfunction is formed from the 'don't care' terms of the Boolean function. The subfunctions are represented on separate Karnaugh maps and these are independently transformed to RM coefficient maps. Each of the Boolean 'don't care' terms may be assigned values (either 0 or 1) until each cell of the equivalent RM coefficient map contains a binary value. The contents of both RM coefficient maps may then be summed (modulo-2). The resulting RM coefficient map represents the FPRM expansion of the original Boolean function where the 'don't care' terms have been assigned values. Green proposed determining the optimum FPRM expansion of the initial Boolean function through exhaustive search. That is, deriving RM coefficient maps for all possible combinations of values for the 'don't care' terms. Each RM coefficient map would in turn be added to the RM coefficient map representing the specified terms of the initial Boolean function. The resulting RM coefficient map which displays the fewest number of cell containing is represents the minimal FPRM expansion. If the Boolean function is comprised of $k$ 'don't care' terms then $2^{k} \mathrm{RM}$ coefficient maps will be generated.

Habib [40, 41] employed $1 \times 2^{n}$ Boolean matrices to represent $n$ variable Boolean functions, FPRM expansions and ESOP forms. This type of notation allows the representation of incompletely specified Boolean functions. Habib illustrated a heuristic technique for deriving FPRM expansions and ESOP forms from Boolean functions with 'don't care' terms. The technique utilises the matrix folding method previously outlined in section 3.1 of chapter 3. The matrix is partition into equal parts according to the state of any function variable. The 'don't care' terms are then assigned values which minimise the number of 1 s in the matrix formed from the modulo-2 sum of the components of both partitions of the original matrix. This operation is repeated for all function variables. It cannot be guaranteed that this method will determine the optimum allocation of the 'don't care' terms. Indeed, the quality of the solution is affected by the order in which the partitioned matrices are formed. The technique is suitable for transforming incompletely specified Boolean functions to FPRM expansions of pre-
determined polarity and may also be used in conjunction with the heuristic minimisation techniques for determining minimal (sub-minimal) FPRM expansions [41] and ESOP forms [40]. (Note, in all of the techniques described the only arithmetic operation performed is modulo-2 addition, matrix multiplication is not employed.)

Harking [42] presented a method for deriving a $2^{n} \times 2^{n}$ polarity matrix representing all $2^{n}$ FPRM expansions of a $n$ variable Boolean function. The Boolean matrix notation is employed and modulo-2 addition, not matrix multiplication, is used to form the polarity matrix. This technique may be used to determine the optimum (sub-optimum) FPRM expansions representing an incompletely specified Boolean function. The initial Boolean function is partitioned into 2 parts, specified minterms and 'don't care' terms. The $2^{n} \times 2^{n}$ polarity matrix representing all $2^{n}$ FPRM expansions is then constructed from the Boolean matrix representing the specified terms of the function. The 'don't care' terms are used to form a $2^{n} x 2^{n}$ Boolean matrix which represents the effects of these 'don't care' terms on the coefficients of each FPRM expansion. Harking then formulated a technique which alters the relevant coefficients of each FPRM expansion according to the value of a 'don't care' term. The technique may be employed to determine all $2^{n}$ FPRM expansions for all combinations of values of 'don't care' terms. Thus the optimum FPRM expansions of the initial incompletely specified Boolean function may be determined through exhaustive search. Alternatively, two heuristic techniques are presented, both lead to a reduction in the number of FPRM expansions which must be evaluated but the quality of the solution is obviously degraded.

A novel heuristic technique for deriving optimal (sub-optimal) polarity 0 FPRM expansions of incompletely specified Boolean functions was introduced by Varma and Trachtenberg [62]. This method may be used with a Boolean matrix representation where Boolean functions are transformed to the equivalent RM expansion using matrix multiplication (modulo-2 multiplication and addition are performed). Alternatively, equations are detailed which may be employed to compute the coefficients of RM expansions. Initially, the equivalent RM expansion of the incompletely specified Boolean function with all 'don't care' terms equated to 0 is generated. The effects of the 'don't care' terms on the terms of the RM expansion are then evaluated. The first
'don't care' term is considered and a value is allocated so as to minimise the number of terms in the RM expansion derived from the Boolean function with all unspecified terms set equal to 0 . Hence, the RM expansion is altered so as to represent the Boolean function with the 'don't care' term assuming the value determined in the previous step. Another 'don't care' term is then allocated a value, the aim being to further reduce the number of terms in the new RM expansion. This RM expansion is now altered according to the effects of the 'don't care' term. The process is repeated until all 'don't care' have been assigned values. The resulting RM expansion is a minimal (sub-minimal) representation of the original incompletely specified Boolean function. It is of interest to note that Varma and Trachtenberg use the numbers of literals in a RM expansion as a measure of the complexity of the implementation. Thus, a cost function is introduced into their algorithm. Additionally, Varma and Trachtenberg [62] determine limits for the maximum number of product terms in the RM expansion representing an incompletely specified Boolean function. Finally, the heuristic technique is extended to allow reduced representations to be used when deriving minimal (sub-minimal) FPRM expansions of incompletely specified Boolean functions.

### 4.3 Minimisation of Fixed Polarity Reed-Muller Expansions using Unspecified Product Terms

The technique described in this section determines the allocation of the unspecified coefficients of a Boolean function which results in a minimal FPRM expansion, where the polarity of the FPRM expansion is predetermined. It has been stated that the coefficients $b_{k}\left(k=0,1, \ldots, 2^{n}-1\right)$ of any FPRM expansion depend on the coefficients $d_{i}$ of the equivalent Boolean function ( $i=0,1, \ldots, 2^{n}-1$ ). The value of coefficients $b_{k}$ will depend on the coefficients $d_{i}$, which take the values 0,1 or D. Henceforth, $d_{i}$ will denote the specified coefficients of the Boolean function which take the value 1. Any coefficient which is unspecified and so takes the value $D$ will be denoted $d_{k, r}$ where $d_{k, r} \in\{0,1\}$, $i, k \in\left\{0,1, \ldots, 2^{n}-1\right\}$, $i \neq k$, and $r=$ $0,1, \ldots, t$ where $t$ is the number of unspecified terms in the Boolean function.

In order to evaluate the FPRM coefficients, $b_{k^{\prime}}$, it is necessary to establish some rules,

$$
\begin{equation*}
d_{i} \oplus d_{k}=(1 \oplus 1)=0 \text { (terms cancel) } \tag{i}
\end{equation*}
$$

(ii) $\quad d_{k, s} \oplus d_{k, s}=0$ (terms cancel)
(iii) $d_{i} \oplus d_{k, r}=1 \oplus d_{k, r}$
(iv) $\quad d_{i} \oplus\left(d_{k, r} \oplus d_{l, r+1} \oplus \ldots \oplus d_{m, s}\right)=1 \oplus d_{k, r} \oplus d_{l, r+1} \oplus \ldots \oplus d_{m, s}$
$i, k, \ell, m \in\left\{0,1, \ldots, 2^{n}-1\right\}, i \neq k \neq \ell \neq m$
$r, s \in\{1, \ldots, t\}, r \neq s$

It is a simple task to derive the coefficients $b_{k}$ which are independent of any coefficient $d_{l, r}$. These coefficients are termed the specified FPRM expansion coefficients and may take the value 0 or 1 . The remaining $b_{k}$ coefficients are dependent on $d_{\ell, r}$ and are the unspecified FPRM expansion coefficients. The minimisation problem is, therefore, to allocate the coefficients $d_{\ell, r}$ such that a minimal number of coefficients $b_{k}$ have the value 1 .

One approach to solving this problem is to evaluate each $b_{k}$ for all combinations of $t$ coefficients, $d_{\ell, r}$, being set to 0 and to 1 , where each $b_{k}$ is dependent on some $d_{\ell, r^{\circ}}$ Alternatively, a technique which offers the possibility of determining optimum assignments of $d_{\ell, r}$ without performing an exhaustive search is now proposed. This technique identifies combinations of coefficients, $d_{\ell, r}=0$ or 1 , which cannot lead to optimum allocations. The procedure employs a tree-type structure (Figure 4.2) similar to the BDDs presented in chapter 2.

Each node (box) has one input branch and 2 output branches. The left output branch denotes a path on which the value of coefficient $d_{m, s}$ is 0 , whilst the value of $d_{a, s}$ on the right output branch is 1. Each node contains a value denoted score which is equal to the number of coefficients $b_{k}$ which equal 0 when the coefficients $d_{m, s}$ assume the values indicated on the branches leading from the top of the tree to the node. The initial value of score is 0. A group of FPRM expansion coefficients, $b_{k}$, may be evaluated at each level of the tree and the scores are carried down from one level to the next. The total number of levels in the tree is equal to $t_{1}$ the number of Boolean 'don't care' terms. The maximum score at the final level (maxscore) is the maximum number of FPRM expansion coefficients, $b_{k}$, which can be set to zero, and the path leading to maxscore indicates the optimum allocation of the Boolean 'don't care' terms. It is possible to reduce the number of calculations which must be undertaken to find a solution. This


Figure 4.2: Tree-type structure used to determine the optimum use of 'don't care' terms.
means that the optimum allocation of Boolean 'don't care' terms can be found without performing an exhaustive search. The technique is now explained. The unspecified FPRM expansion coefficients should be partitioned into groups according to the coefficients $d_{i, r}$ on which they depend,
i.e.

Group 1 coefficients $b_{k}$ dependent on $d_{1,1}$ only.
Group 2 coefficients $b_{\ell}$ dependent on $d_{k, 2}$ or $\left(d_{i, 1} \oplus d_{k, 2}\right)$.
Group 3 coefficients $b_{m}$ dependent on $d_{l, 3},\left(d_{i, 1} \oplus d_{l, 3}\right),\left(d_{k, 2} \oplus d_{l, 3}\right)$ or $\left(d_{1,1} \oplus d_{k, 2} \oplus d_{l, 3}\right)$.

Group $t \quad$ coefficients $b_{q}$ dependent on $d_{s, t}\left(d_{t, 1} \oplus d_{s, t}\right),\left(d_{k, 2} \oplus d_{s, t}\right)$, $\left(d_{1,1} \oplus d_{k, 2} \oplus d_{s, t}\right),\left(d_{l, 3} \oplus d_{s, t}\right), \ldots \ldots$, $\left(d_{t, 1} \oplus d_{k, 2} \oplus \ldots \oplus d_{m,(t-2)} \oplus d_{s, t}\right)$ $\left(\right.$ or $\left.d_{i, 1} \oplus d_{k, 2} \oplus \ldots \oplus d_{p,(t-1)} \oplus d_{s, t}\right)$
$i, k, \ell, m, p, q \in\left\{0,1, \ldots, 2^{n}-1\right\}$

The number of FPRM coefficients in each of these groups should be counted. Let $d c_{1}$ equal the number of coefficients, $b_{k^{\prime}}$, which depend on only $d_{i, 1}$ and, hence, are in Group 1. Then, $d c_{2}$ will denote the number of unspecified FPRM expansion coefficients in Group 2, and so on. Let $d c_{\text {tot }}$
equal the total number of unspecified FPRM expansion coefficients, then $d c_{t o t}=d c_{1}+d c_{2}+\ldots+d c_{t}$. Additionally, $d c$ equals the number of FPRM expansion coefficients which at any stage of the allocation procedure have not been evaluated. Considering the structure illustrated in Figure 4.2, as the coefficients $b_{k}$ are evaluated a score is calculated. Initially, score is set equal to 0 , then score is the total number of coefficients $b_{k}$ which can be equated to zero when the values of all $d_{m}, s$ assume the values indicated on the path leading to the node associated with score. At any level there may be a range of scores. An inequality may be formulated which makes it possible to determine which paths of the structure may be terminated at a level prior to the final level, thus reducing the number of calculations required to find an optimum allocation of the coefficients $d_{m, s}$. Hence, at any level, a path which leads to a node with value score should be terminated when

$$
\text { score }+d c<\operatorname{maxscore}+\left\lceil\frac{d c}{2}\right\rceil
$$

where score is any score at level $\ell$, maxscore is the maximum score at level $l$, and $d c$ is the number of coefficients $b_{k}$ which have yet to be evaluated. ( $\lceil x\rangle$ is the smallest integer greater than or equal to the real number $x$ and $\lfloor x\rfloor$ is the greatest integer less than or equal to the real number $x$.)

The validity of this inequality is now proved.

## Proof

At level $\ell(\ell=1,2, \ldots, t)$ assume that $d c$ FPRM expansion coefficients, $b_{k}$, are yet to be evaluated and the maximum score is maxscore ( $d c$ and maxscore have integer values). Consider evaluating all dc FPRM expansion coefficients at level $\ell+1$, i.e. all remaining $b_{k}$ coefficients are dependent on $d_{i, \ell+1}$ and some combination of coefficients $d_{k, 1}, d_{m, 2}, \ldots, d_{p, l}$. Two new scores will be derived for any score at level $\ell$. One new score will indicate the number of $b_{k}$ coefficients equated to 0 when $d_{1, l_{1}}=0$, whilst the other will indicate how many coefficients are equated to 0 when $d_{1, \ell_{+1}}=1$ (coefficients $d_{k, 1}, d_{m, 2}, \ldots, d_{p, l}$ assume the values indicated on the branches leading to the node with value score). If all $b_{k}$ coefficients are equated to 0 when $d_{1, \ell+1}=0$ (or $d_{i, \ell_{+1}}=1$ ) then the new values of score, i.e. the values at level $\ell+1$, will be score and (score $+d c$ ). Alternatively, if only
half the coefficients are equated to 0 when $d_{1, \ell_{1}}=0$ (or $d_{1, \ell+1}=1$ ) then the new values of score will be

$$
\left(\text { score }+\left\lceil\frac{d c}{2}\right\rceil\right) \text { and }\left(\text { score }+\left\lfloor\frac{d c}{2}\right\rfloor\right) \cdot\left(\text { Note } d c=\left\lceil\frac{d c}{2}\right\rceil+\left\lfloor\frac{d c}{2}\right\rfloor\right)
$$

It is now possible to establish bounds on the scores at level $\ell+1$.
Upper limit on the maximum score at level $\ell+1$ is
(maxscore + dc)
and the lower limit on maximum score at level $\ell+1$ is

$$
\left(\operatorname{maxscore}+\left\lceil\frac{d c}{2}\right\rceil\right)
$$

Upper limit on any score at level $\ell+1$ is

$$
(\text { score }+d c)
$$

Therefore, if any score at level $\ell$ has a value such that

$$
\text { score }+d c<\operatorname{maxscore}+\left\lceil\frac{d c}{2}\right\rceil
$$

then the path emanating from that node cannot lead to a maximum score at level $\ell+1$.

This inequality is also valid for the case where dc coefficients are allocated not at level $\ell+1$ but at levels $\ell+1, \ell+2, \ldots, \ell+s$. (Note, $t=\ell+s$.) The $b_{k}$ FPRM expansion coefficients are grouped according to their dependency on the coefficients $d_{i, r}$ and the number of coefficients in each group is indicated by the values $d c_{l_{1}}, d c_{l_{2}}, \ldots, d c_{t}$. It will be shown that the established limits are valid for this case. Once again, let score denote the value of any score at level $l$ and maxscore be the maximum score at level $\ell$.
Upper limit on maximum score at level $\ell+s$ is

$$
\left(\text { maxscore }+d c_{l+1}+d c_{l+2}+\ldots+d c_{l+s}\right)
$$

and the lower limit on maximum score at level $\ell+s$ is

$$
\left(\text { score }+\left\lceil\frac{d c_{l+1}}{2}\right\rceil+\left\lceil\frac{d c_{l+2}}{2}\right\rceil+\ldots . .+\left\lceil\frac{d c_{l+s}}{2}\right\rceil\right)
$$

Upper limit on any score at level $\ell+s$ is

$$
\left(\text { score }+d q_{+1}+d q_{+2}+\ldots+d q_{+s}\right)
$$

Therefore, if any score at level $\ell+S$ has a value such that

$$
\text { score }+d c_{l+1}+d c_{l+2}+\ldots+d c_{l+s}<\operatorname{maxscore}+\left\lceil\frac{d c_{l+1}}{2}\right\rceil+\left\lceil\frac{d c_{l+2}}{2}\right\rceil+\ldots+\left\lceil\frac{d c_{l+s}}{2}\right\rceil
$$

then the path cannot lead to a maximum score at level $\ell+s$.
Now, $d c=d q_{+1}+d q_{+2}+\ldots+d q_{+s}$ and
$\left\lceil\frac{d c}{2}\right\rceil \leqslant\left\lceil\frac{d c_{l+1}}{2}\right\rceil+\left\lceil\frac{d c_{l+2}}{2}\right\rceil+\ldots . . .\left\lceil\frac{d c_{l+s}}{2}\right\rceil$,
hence, $($ score $+d c)=\left(\right.$ score $\left.+d c_{l+1}+d c_{l+2}+\ldots+d c_{l_{+s}}\right)$ and $\left(\right.$ score $\left.+\left\lceil\frac{d c}{2}\right\rceil\right) \leq\left(\right.$ score $\left.+\left\lceil\frac{d c_{l+1}}{2}\right\rceil+\left\lceil\frac{d c_{l+2}}{2}\right\rceil+\ldots . . .\left\lceil\frac{d c_{l+s}}{2}\right\rceil\right)$

Thus the validity of the inequality is confirmed.

Any node with value score which satisfies the following inequality will not lead to a node with value maxscore at level $t$.

$$
\text { score }+d c<\operatorname{maxscore}+\left\lceil\frac{d c}{2}\right\rceil
$$

(End of Proof)

The following algorithm employs the techniques introduced in the preceding discussion to provide a means of determining the optimum use of the 'don't care' minterms of a Boolean function. The resulting expression is a minimal FPRM expansion of the initial Boolean function. The following terms are used in the algorithm and therefore require to be formally defined. The term dc_count denotes the number of unspecified FPRM expansion coefficients $b_{k}$ which have been evaluated at level $r$, where $r=1,2, \ldots, t$. score 0 and scorel denote the number of coefficients $b_{k}$ at any level $r$ which are equated to zero when $d_{i, r}=0$ or $d_{i, r}=1$, respectively. score is the total number of coefficients $b_{k}$ which can be equated to zero when the values of all $d_{i, r}$ assume the values indicated on the path leading to the node associated with score.
Maxscore is the maximum value of score at level $r$.
All other terms are as previously defined in this section.

Converting an incompletely specified Boolean function to polarity $p$ FPRM expansion.
S. 1 For $r=1,2, \ldots, t$ denote each coefficient $d_{i}$ of a 'don't care' term of the incompletely specified Boolean function $d_{1, r}\left(i \in\left\{0,1, \ldots, 2^{n}-1\right\}\right)$.
S. 2 Convert the Boolean function to the polarity $p$ FPRM expansion, transforming the 'don't care' terms according to rules (i) - (iv) detailed previously.
S. 3 Form $t$ groups comprised of the unspecified FPRM expansion coefficients $b_{k}$. Group 1 contains coefficients dependent on $d_{1,1}$, Group 2 contains product terms dependent on $d_{k, 2}$ and $\left(d_{i, 1} \oplus d_{k, 2}\right)$, Group 3 contains ... etc..
For $r=1,2, \ldots, t$, let $d c_{r}$ equal the number of coefficients $b_{k}$ in Group $r$. Let $d c_{t o t}=d c_{1}+d c_{2}+\ldots+d c_{t}$ and let $d c=d c_{t o t}$. Set $r=1$.
S. 4 Form level $r$ of the tree structure (see Figure 4.2). If $r \neq 1$ then $S .5$ else construct a left branch denoted $d_{i, r}=0$ and a right branch denoted $d_{i, r}=1$. Each branch terminates in a node. Let the value contained in the node be denoted score. Let score = score0 $=$ score1 $=0$. Go to S.6.
S. 5 Select a level $r-1$ node which is unterminated. Let the value contained in this node be score and from this node construct a left branch denoted $d_{i, r}=0$ and a right branch denoted $d_{i, r}=1$. Each branch terminates in a level $r$ node. Let the value contained in this new node be score where score at level $r$ is equal to score at level $r-1$. Let score0 $=$ score $1=$ score.
S. 6 Let $d c_{-}$count $=d c_{r}$. If $d c_{-}$count $=0$ then S.10. Select the first coefficient $b_{k}$ from Group r. Let $d_{i, r}=0$, also $d_{i, 1}, \ldots, d_{i, r-1}$ assume the values on the branches leading to this node. If $b_{k}=0$ then score0 $=$ score $0+1$ else score1 $=$ score $1+1$. Let dc_count $=$ dc_count-1.
S. 7 If dc_count $=0$ then S. 10 else S.8.
S. 8 Select next coefficient $b_{k}$ from Group r. Let $d_{1, r}=0$, also $d_{1,1}, \ldots$, $d_{i, r-1}$ assume the values on the branches leading to this node. If $b_{k}$ $=0$ then score $0=$ score $0+1$ else score $1=$ score $1+1$. Let dc_count $=d c_{-}$count -1 .
S. 9 If dc_count $=0$ then S. 10 else S.8.
S. 10 Let the score associated with the node attached to branch denoted $d_{1, r}=0$ equal score0, and the score associated with the node
attached to branch denoted $d_{1, r}=1$ equal scorel.
S. 11 Repeat S. 5 to S. 10 until all unterminated level $r-1$ nodes have been selected.
S. 12 Let $d c=d c-d c_{r}$, if $d c=0$ then S.14. Otherwise, denote the value associated with each level $r$ node as score. Find maxscore at level $r$. If any node has

$$
\text { score }+d c<\operatorname{maxscore}+\left\lceil\frac{d c}{2}\right\rceil
$$

then terminate the path at this node.
S. 13 Let $r=r+1$. Go to S.5.
S. 14 Denote the value associated with each level $t$ node as score. Find maxscore at level $t$.

The values $d_{i, r}=0$ and $d_{1, r}=1, r=1,2, \ldots, t$ on the paths leading to the level $t$ nodes with value maxscore indicate the allocation of the 'don't care' terms which leads to an optimum FPRM expansion.
S. 15 The optimum FPRM expansions may be derived by substituting the appropriate sets of values of $d_{i, r}$ into the incompletely specified FPRM expansion.

### 4.4 Tabular Technique for Deriving Fixed Polarity Reed-Muller Expansions from Incompletely Specified Boolean Functions

The tabular technique reviewed in chapter 3 section 3.2 .1 may be employed to transform a completely specified Boolean function to a FPRM expansion. It is possible to extend this procedure to incompletely specified Boolean functions. The method now proposed transforms an incompletely specified Boolean function to an equivalent incompletely specified FPRM expansion.

A specified minterm, $m_{i},\left(i=0,1, \ldots, 2^{n}-1\right)$ of a $n$ variable incompletely specified Boolean function may be represented by the binary $n$-tuple $<i_{n} i_{n-1}$ $\ldots i_{1}>i_{j} \in\{0,1\}, j=1,2, \ldots, n$. Let an unspecified minterm, $m_{k}$, of an incompletely specified Boolean function with $t$ 'don't care' minterms be represented by the binary $n$-tuple $<k_{n} k_{n-1} \ldots k_{1}>d_{k, r}, k_{j} \in\{0,1\}, k \in$ $\left\{0,1, \ldots, 2^{n}-1\right\}, j=1,2, \ldots, n$ and $r=1,2, \ldots, t$. Thus, an incompletely specified Boolean function may be represented by a table of minterms.

Example 4.1 Display the following incompletely specified Boolean function ( $n$ $=3$ ) using the tabular notation
$f\left(x_{3}, x_{2}, x_{1}\right)=\bar{x}_{3} \bar{x}_{2} \bar{x}_{1}+\bar{x}_{3} x_{2} x_{1}+x_{3} \bar{x}_{2} \bar{x}_{1}+D x_{3} \bar{x}_{2} x_{1}+D x_{3} x_{2} \bar{x}_{1}+D x_{3} x_{2} x_{1}$
$D \in\{0,1\}$
Boolean function

| $x_{3}$ | $x_{2}$ | $x_{1}$ |  |
| :--- | :--- | :--- | :--- |
| 0 | 0 | 0 |  |
| 0 | 1 | 1 |  |
| 1 | 0 | 0 |  |
| 1 | 0 | 1 | $d_{5,1}$ |
| 1 | 1 | 0 | $d_{6,2}$ |
| 1 | 1 | 1 | $d_{7,3}$ |

## (End of example)

Now consider transforming a incompletely specified Boolean function to the equivalent PPRM expansion using a modified form of the original tabular technique [1] and the tabular notation just described. Let $\Delta_{v}=\oplus \Sigma \delta$, where $\delta \in\left\{1, d_{f, 1}, d_{k, 2}, \ldots, d_{m, t}\right\}, i, k, m, v \in\left\{0,1, \ldots, 2^{n}-1\right\}$. If any term in the table has the form $\left\langle i_{n} \ldots i_{j+1} 0 i_{j-1} \ldots i_{1}\right\rangle\left(\left\langle i_{n} \ldots i_{j+1} 0 i_{j-1} \ldots i_{1}>\Delta_{v}\right)\right.$ then generate a new term which is represented by $\left\langle i_{n} \ldots i_{j+1} 1 i_{j-1} \ldots i_{1}\right\rangle\left(\left\langle i_{n} \ldots i_{j+1} 1 i_{j-1} \ldots i_{1}>\Delta_{V}\right)\right.$. The newly generated terms are then compared with the existing terms. If a term of the existing expansion (table) and a generated term are found to satisfy any of the cases listed below then both terms are deleted and a new term is formed (resulting term). This resulting term is then added to the existing expansion (table).

## Existing term (Generated term) <br> $<i_{n} i_{n-1} \ldots i_{1}>$

$<i_{n} i_{n-1} \ldots i_{1}>d_{i, r} \quad<i_{n} i_{n-1} \ldots i_{1}>d_{1, r}$

Resulting term

$$
\begin{aligned}
& \left\langle i_{n} i_{n-1} \cdots i_{1}\right\rangle(1 \oplus 1) \\
& =\left\langle i_{n} i_{n-1} \cdots i_{1}\right\rangle(0)=0
\end{aligned}
$$

$$
\left\langle i_{n} i_{n-1} \ldots i_{1}\right\rangle\left(d_{1, r} \oplus d_{i, r}\right)
$$

$$
=\left\langle i_{n} i_{n-1} \cdots i_{1}\right\rangle(0)=0
$$

$$
\begin{array}{lll}
<i_{n} i_{n-1} \ldots i_{1}> & <i_{n} i_{n-1} \ldots i_{1}>d_{1, r} & <i_{n} i_{n-1} \ldots i_{1}>\left(1 \oplus d_{1, r}\right) \\
<i_{n} i_{n-1} \ldots i_{1}>d_{1, r} & <i_{n} i_{n-1} \ldots i_{1}>d_{k, s} & <i_{n} i_{n-1} \ldots i_{1}>\left(d_{1, r} \oplus d_{k, s}\right)
\end{array}
$$

In general,
$<i_{n} i_{n-1} \ldots i_{1}>\Delta_{v} \quad<i_{n} i_{n-1} \ldots i_{1}>\Delta_{W} \quad<i_{n} i_{n-1} \ldots i_{1}>\left(\Delta_{v} \oplus \Delta_{W}\right)$
where $\Delta_{v}=\oplus \Sigma \delta, \Delta_{w}=\oplus \Sigma \delta, \delta \in\left\{1, d_{1,1}, d_{k, 2}, \ldots, d_{m, t}\right\}, v, w \in\left\{0,1, \ldots, 2^{n}-1\right\}$

The procedure is repeated, generating new terms for variable $\boldsymbol{x}_{\boldsymbol{k}}, k \in$ $\{1,2, \ldots, n), k \neq j$, and updating the expansion according to the above rules formulated for the tabular technique. The conversion is complete when the procedure has been applied for each expansion variable. The resulting expansion is the polarity 0 FPRM expansion.

It is possible to convert the RM expansion to another FPRM expansion. The technique detailed for completely specified Boolean functions may be employed (section 3.2 .1 of chapter 3). Additionally, the rules formulated for transforming the 'don't care' terms should be applied. The resulting expression is a FPRM expansion comprised of specified and unspecified product terms. The optimum allocation of the unspecified terms may be derived using the algorithm detailed in section 4.3.

The following example illustrates the use of the algorithm detailed in section 4.3 to determine the optimum allocation of the 'don't care' terms of a incompletely specified Boolean function. The tabular notation is employed to represent the function.

Example 4.2 Determine the minimum (sub-minimum) polarity 2 FPRM expansion of the incompletely specified Boolean function

$$
f\left(x_{4}, x_{3}, x_{2}, x_{1}\right)=\sum m(1,4,7,8,9,11)+\sum d(3,5,6,15)
$$

S. 1 Denote the unspecified minterms of the Boolean function $d_{3,1}, \ldots, d_{15,4}$ Boolean function

| $x_{4}$ | $x_{3}$ | $x_{2}$ | $x_{1}$ |  |
| :--- | :--- | :--- | :--- | :--- |
| 0 | 0 | 0 | 1 |  |
| 0 | 1 | 0 | 0 |  |
| 0 | 1 | 1 | 1 |  |
| 1 | 0 | 0 | 0 |  |
| 1 | 0 | 0 | 1 |  |
| 1 | 0 | 1 | 1 |  |
| 0 | 0 | 1 | 1 | $d_{3.1}$ |
| 0 | 1 | 0 | 1 | $d_{5,2}$ |
| 0 | 1 | 1 | 0 | $d_{6,3}$ |
| 1 | 1 | 1 | 1 | $d_{15.4}$ |

S. 2 Transform the Boolean function to the polarity 0 FPRM expansion.

Polarity 0 FPRM expansion

|  | $x_{4}$ | $x_{3}$ | $x_{2}$ | $x_{1}$ |  |
| :--- | :--- | :--- | :--- | :--- | :--- |
| ${ }^{*}$ | 0 | 0 | 0 | 1 |  |
| ${ }^{*}$ | 0 | 1 | 0 | 0 |  |
|  | 0 | 1 | 1 | 1 | $1 \oplus d_{3,1}{ }^{\oplus} d_{5,2} \oplus d_{6,3}$ |

$\begin{array}{lllll}* & 1 & 0 & 0 & 0\end{array}$
$\begin{array}{lllll}1 & 0 & 1 & 1 & d_{3,1}\end{array}$
$\begin{array}{lllll}0 & 0 & 1 & 1 & 1 \oplus d_{3,1}\end{array}$
$*_{4} \quad 0 \quad 1 \quad 0 \quad 1 \quad d_{5.2}$
$\begin{array}{lllll}0 & 1 & 1 & 0 & 1 \oplus d_{6,3}\end{array}$
$1111 \quad 1 \quad d_{3,1} \oplus d_{5.2} \oplus d_{6,3} \oplus d_{15,4}$
$*_{5} \quad 1 \quad 1 \quad 1 \quad 0 \quad 1 \quad d_{5,2}$
$1 \quad 1 \quad 1 \quad 0 \quad d_{6,3}$
$\begin{array}{llll}1 & 0 & 1 & 0\end{array}$
$\begin{array}{lllll}* & 1 & 0 & 0 & 1\end{array}$
S. 2 Transform the Boolean function to the polarity 2 FPRM expansion, applying the transformation rules.
( ${ }_{i}$ denotes existing terms and generated terms which are equivalent)

$$
\begin{aligned}
& \text { Generated terms } \\
& \left(\bar{x}_{2}\right) \\
& *_{4} \frac{x_{4}}{} \begin{array}{llll}
x_{3} & \bar{x}_{2} & x_{1} \\
\hline 0 & 1 & 0 & 1
\end{array} 1 \oplus d_{3,1} \oplus d_{5,2} \oplus d_{6,3} \\
& \begin{array}{llllll}
*_{6} & 1 & 0 & 0 & 1 & d_{3,1}
\end{array} \\
& { }^{*}{ }_{1} \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \oplus d_{3,1} \\
& *_{2} \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \oplus d_{6,3} \\
& { }^{*}{ }_{5} \quad 1 \quad 1 \quad 0 \quad 1 \quad d_{3,1} \oplus d_{5,2} \oplus d_{6,3} \oplus d_{15,4} \\
& \begin{array}{lllll}
1 & 1 & 0 & 0 & d_{6,3}
\end{array} \\
& \begin{array}{lllll}
* & 1 & 0 & 0 & 0
\end{array}
\end{aligned}
$$

Polarity 2 FPRM expansion

| $x_{4}$ | $x_{3}$ | $\bar{x}_{2}$ | $x_{1}$ |  |
| :--- | :--- | :--- | :--- | :--- |
| 0 | 0 | 0 | 1 |  |
| 0 | 1 | 0 | 0 | $d_{3,1}$ |
| 0 | 1 | 1 | 1 | $1 \oplus d_{3,1} \oplus d_{5,2} \oplus d_{6,3}$ |
| 1 | 0 | 1 | 1 | $d_{3,1}$ |
| 0 | 0 | 1 | 1 | $1 \oplus d_{3,1}$ |
| 0 | 1 | 0 | 1 | $1 \oplus d_{3,1} \oplus d_{6,3}$ |
| 0 | 1 | 1 | 0 | $1 \oplus d_{6,3}$ |
| 1 | 1 | 1 | 1 | $d_{3,1} \oplus d_{5,2} \oplus d_{6,3} \oplus d_{15,4}$ |
| 1 | 1 | 0 | 1 | $d_{3,1} \oplus d_{6,3} \oplus d_{15,4}$ |
| 1 | 1 | 1 | 0 | $d_{6,3}$ |
| 1 | 0 | 1 | 0 |  |
| 1 | 0 | 0 | 1 | $1 \oplus d_{3,1}$ |
| 1 | 1 | 0 | 0 | $d_{6,3}$ |

The incompletely specified Boolean function has been transformed to the equivalent polarity 2 FPRM expansion.
S. 3 Group the product terms according to the unspecified minterms on which they depend.

Unspecified FPRM terms
Group $1 \quad d_{3,1}, d_{3,1},\left(1 \oplus d_{3,1}\right),\left(1 \oplus d_{3,1}\right)$
Group 2 No unspecified FPRM terms are

Group 3
dependent only on $d_{5,2}$, or $\left(d_{3,1} \oplus d_{5,2}\right)$
Group $3 \quad d_{6,3}, d_{6,3}, d_{6,3},\left(1 \oplus d_{6,3}\right)$,
$d c_{3}=6$
$\left(1 \oplus d_{3,1} \oplus d_{6,3}\right),\left(1 \oplus d_{3,1} \oplus d_{5,2} \oplus d_{6,3}\right)$
Group $4 \quad\left(d_{3,1} \oplus d_{6,3} \oplus d_{15,4}\right)$,
$d c_{4}=2$
$\left(d_{3,1} \oplus d_{5,2} \oplus d_{6,3} \oplus d_{15,4}\right)$

Total number of FPRM expansion unspecified terms is $12, d c_{t o t}=12$
S. 4 - S. 13 Set $d_{3.1}$ first to 0 and then to 1 and evaluate the effects on the 'don't care' coefficients of the FPRM expansion. This is illustrated in Figure 4.3. Additionally, determine which combinations of Boolean 'don't care' coefficients need not be evaluated. Repeat for $d_{5,2}, d_{6,3}$ and $d_{15,4^{\circ}}$.
$d c=d c_{t o t}=12$
$d c=d c-d c_{1}=12-4=8$
maxscore $=2$
No paths can be terminated
$d c=d c-d c_{2}=8-0=8$
maxscore $=2$
No paths can be terminated
$d c=d c-d c_{3}=8-6=2$
maxscore $=7$
Terminate all paths with
level 3 nodes containing
score < 6
$d c=d c-d c_{4}=2-2=0$
maxscore $=9$


Figure 4.3: Determining the optimum use of the 'don't care' terms for Example 4.2.

Note: At level 3 maxscore $=7$ and $d c=2$,
Evaluating the inequality

$$
\begin{aligned}
& \text { score }+d c<\text { maxscore }+\left\lceil\frac{d c}{2}\right\rceil \\
& \text { score }+2<7+1
\end{aligned}
$$

determines that any node with score < 6 should be terminated.
S. 14 At level 4 maxscore $=9$, hence all paths which terminate in nodes with score $=9$ indicate the optimum allocation of Boolean 'don't care' terms, i.e. $d_{3.1}=d_{15,4}=1, d_{5.2}=d_{6.3}=0$.
S. 15 Substitute this assignment of $d_{3,1}, \ldots, d_{15,4}$ into the incompletely specified polarity 2 FPRM expansion. This generates the minimal polarity 2 FPRM expansion of the incompletely specified Boolean function.
Optimum polarity 2 FPRM expansion
$f_{2}\left(x_{4}, x_{3}, x_{2}, x_{1}\right)=x_{3} \bar{x}_{2} \oplus x_{4} \bar{x}_{2} \oplus x_{1} \oplus x_{4} \bar{x}_{2} x_{1}$

| $x_{4}$ | $x_{3}$ | $\bar{x}_{2}$ | $x_{1}$ |
| :--- | :--- | :--- | :--- |
| 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |

Note, this is derived from the completely specified Boolean function

$$
\begin{aligned}
f\left(x_{4}, x_{3}, x_{2}, x_{1}\right) & =\sum m(1,4,7,8,9,11,3,15) \\
& =\bar{x}_{4} \bar{x}_{3} \bar{x}_{2} x_{1}+\bar{x}_{4} x_{3} \bar{x}_{2} \bar{x}_{1}+\bar{x}_{4} x_{3} x_{2} x_{1}+x_{4} \bar{x}_{3} \bar{x}_{2} \bar{x}_{1} \\
& +x_{4} \bar{x}_{3} \bar{x}_{2} x_{1}+x_{4} \bar{x}_{3} x_{2} x_{1}+\bar{x}_{4} \bar{x}_{3} x_{2} x_{1}+x_{4} x_{3} x_{2} x_{1}
\end{aligned}
$$

It is of interest to note that the optimum allocation of the 'don't care' terms in the Boolean domain is $d_{3,1}=d_{5,2}=d_{6,3}=1, d_{15,4}=0$.

### 4.5 Determining Minimal FPRM Expansions of Incompletely Specified Boolean Functions

Determining minimal FPRM expansions of an incompletely specified Boolean function is a complex problem. It combines the task of finding the optimum assignment of 'don't care' terms with the search for the minimum FPRM expansions representing the function. Any $n$ variable Boolean function may be represented by a total of $2^{n}$ FPRM expansions. Additionally, an incompletely specified Boolean function with $t$ 'don't care' terms may be represented by any one of $2^{t}$ fully specified Boolean functions. An exhaustive search would generate $2^{n+t}$ FPRM expansions and is obviously impractical for all functions except those with very few variables and low numbers of unspecified terms.

Another approach to solving the minimisation problem is to consider initially, only the specified minterms of the Boolean function and so determine the minimum FPRM expansions representing the function. The 'don't care' terms may then be allocated so as to further reduce the numbers of product terms in the FPRM expansions. Alternatively, an optimum allocation of the 'don't care' terms may be derived for a FPRM expansion where the polarity is randomly selected. The minimal FPRM expansions may then be determined from this fully specified FPRM expansion. Unfortunately, neither approach can guarantee to identify the minimal FPRM expansion of an incompletely specified Boolean function. However, the number of possible combinations of polarity and 'don't care' terms which must be evaluated when employing either of these approaches is a minimum of $\left(2^{n}+2^{t}\right)$. This quantity is significantly less than the $2^{n+t}$ FPRM expansions generated when performing a full exhaustive search, however, the number grows rapidly as $n$ and $t$ increase. Once again, either of these techniques may be applied only to functions with limited numbers of variables and unspecified terms.

It is, perhaps, necessary to emphasise that only one of the three methods detailed above can be guaranteed to find the minimum FPRM expansions of an incompletely specified Boolean function. Obviously, this is the method which performs the full exhaustive search. Additionally, if both of the heuristic techniques are applied to any Boolean function then two different minimal (sub-minimal) FPRM expansions may be derived. It is possible that
one of these FPRM expansions is the minimal form but this cannot be guaranteed. This is illustrated by the following example.

Example 4.3 Determine the minimum FPRM expansion of the incompletely specified Boolean function

$$
\begin{align*}
f\left(x_{3}, x_{2}, x_{1}\right) & =\sum m(0,2,3,4,7)+\sum d(1,5) \\
& =\bar{x}_{3} \bar{x}_{2} \bar{x}_{1}+\bar{x}_{3} x_{2} \bar{x}_{1}+\bar{x}_{3} x_{2} x_{1}+x_{3} \bar{x}_{2} \bar{x}_{1}+x_{3} x_{2} x_{1}+D \bar{x}_{3} \bar{x}_{2} x_{1}+D x_{3} \bar{x}_{2} x_{1} \tag{4.1}
\end{align*}
$$

Three possible approaches to solving this minimisation problem are now described.
(i) Determine through exhaustive search the optimum FPRM expansion of the completely specified Boolean function, i.e. set $D$ equal to zero for both 'don't care' terms. Then transform the 'don't care' terms to the FPRM expansion where the polarity is that determined in the previous operation. Employ these 'don't care' terms to further reduce the number of terms in the FPRM expansion. (The technique described in section 4.3 may be utilised for this task.)

The polarity 5 FPRM expansion is the minimal representation of the specified term of the Boolean function of Equation (4.1).

$$
f_{5}\left(x_{3}, x_{2}, x_{1}\right)=\bar{x}_{1} \oplus x_{2} \oplus \bar{x}_{3} x_{2} \bar{x}_{1}
$$

The number of terms in the polarity 5 FPRM expansion cannot be further reduced by employing the 'don't care' terms. However, allocating the 'don't care' terms in the following manner ensures that the number of terms in the FPRM expansion does not increase, $d_{1,1}$ $=d_{5,2}=0$ and $d_{1,1}=d_{5,2}=1$.
(ii) Form the optimum polarity 0 FPRM expansion of the incompletely specified Boolean function of Equation (4.1). (The technique described in section 4.3 may be utilised for this task.) Next, determine the optimum FPRM expansion of the polarity 0 FPRM expansion derived in the previous operation.

The optimum use of the 'don't care' terms gives rise to two minimal RM expansions (polarity 0 ).

$$
\begin{align*}
& d_{1,1}=1, d_{5,2}=0 \\
& f_{0}\left(x_{3}, x_{2}, x_{1}\right)=1 \odot x_{3} x_{1} \odot x_{3} x_{2}  \tag{4.2}\\
& d_{1,1}=d_{5,2}=1 .
\end{align*}
$$

The RM expansion of Equation (4.2) may be transformed to the polarity 3 FPRM expansion which also comprises of 3 product terms. These representations are the minimum forms corresponding to the allocation of 'don't care' terms. However, the RM expansion of Equation (4.3) may be transformed to the polarity 1 FPRM expansion which comprises of 2 product terms.

$$
\begin{equation*}
f_{1}\left(x_{3}, x_{2}, x_{1}\right)=1 \oplus x_{3} x_{2} \bar{x}_{1} \tag{4.4}
\end{equation*}
$$

(iii) The optimum FPRM expansion of the incompletely specified Boolean function of Equation (4.1) may be determined by performing a complete exhaustive search. This involves generating all $2^{3}$ FPRM expansions for each combination of 'don't care' terms. Thus, the total number of FPRM expansions which must be generated is $2^{3+2}=32$. The results of this search indicate that the optimum representation is the polarity 1 FPRM expansion (Equation (4.4)) generated from a Boolean function where $d_{1,1}=d_{5,2}=1$.

This example illustrates that the optimum FPRM expansion of an incompletely specified Boolean function can only be determined by performing a complete exhaustive search as described in (iii) above. Additionally, the heuristic technique which is employed influences the quality of the solution. (End of example)

The large numbers of FPRM expansions which must be derived when utilising any of the techniques previously described in this section render these method impractical for Boolean functions with large numbers of
variables and 'don't care' terms. It is, therefore, more realistic to consider heuristic minimisation techniques as a means of solving the minimisation problem within a reasonable time scale. The Full Gains minimisation algorithm (or any of the modified algorithms described in chapter 3) may be employed together with the exact technique for deriving optimum assignments of the 'don't care' terms of an incompletely specified Boolean function (section 4.3). The two heuristic approaches which have been detailed above are now considered using the Full Gains method to determine minimal (sub-minimal) FPRM expansions of the Boolean functions. The 'don't care' terms are allocated using the technique detailed in section 4.3.

Apply Full Gains minimisation algorithm then assign 'don't care' terms. (Full Gains, DC)
S. 1 Employ the Full Gains method to determine the minimum (subminimum) FPRM expansion of the fully specified terms of the Boolean function.
S. 2 The 'don't care' terms are allocated using the technique described in section 4.3, where the polarity of each FPRM expansion is as determined in S.1.
S. 3 The resulting FPRM expansion with the fewest product terms is the minimal (sub-minimal) representation of the initial incompletely specified Boolean function.
This procedure is not ideal as the heuristic Full Gains minimisation algorithm may not have found the minimum FPRM expansions. However, the optimum assignments of 'don't care' terms for the FPRM expansion(s) have been identified.

## Assign 'don't care' terms then apply Full Gains minimisation algorithm. (DC, Full Gains)

S. 1 Transform the initial incompletely specified Boolean function to the equivalent polarity 0 FPRM expansion. Determine the optimum allocation of the 'don't care' terms using the technique described in section 4.3.
S. 2 Employ the Full Gains method to determine the minimum (subminimum) FPRM expansions from the completely specified PPRM expansion.
S. 3 The resulting FPRM expansions are the minimal (sub-minimal)
representations of the initial incompletely specified Boolean function.
In common with the first procedure, this approach can also prove to be unsatisfactory as the optimum allocation of the 'don't care' terms for the PPRM expansion may not be the allocation which leads to a minimal FPRM expansion of different polarity.

The graphs of Figure 4.4 and Figure 4.5 illustrate the effectiveness of both the approaches detailed above. The procedures are identified as Full Gains, DC and DC, Full Gains, indicating the order in which the optimisation algorithms are applied. These techniques are evaluated against a third method, denoted Boolean matrix (DC), which was presented by Habib [41] and is reviewed in section 4.2. The $x$-axis of the graphs of Figure 4.4 and Figure 4.5 indicate the number of variables, specified minterms and 'don't care' terms of an incompletely specified Boolean function. Each algorithm optimised 1000 randomly generated incompletely specified Boolean functions where the numbers of variables and minterms is as indicated on the $x$-axis. The results illustrated in Figure 4.4 indicate the percentage of incompletely specified Boolean functions for which the optimisation algorithm identified a minimal FPRM expansion. (The optimum FPRM expansions of each incompletely specified Boolean function were determined through exhaustive search. That is, for each Boolean function a polarity matrix [42] was generated for every combination of 'don't care' terms.) In Figure 4.5 the graph shows the time taken for each algorithms to optimise each set of 1000 randomly generated incompletely specified Boolean functions and so produce the results illustrated in Figure 4.4. The results indicating user time are given so as to illustrate the time taken by each algorithm to form solutions, relative to one another and not as an absolute quantity. Additionally, the incompletely specified Boolean functions were generated by a random number generator. The output of the random number generator was filtered so as to remove any duplicate minterms. All algorithms (a. Full Gains,DC b. DC, Full Gains c. Boolean matrix (DC) d. exhaustive search (modified form of Harkings' technique [42]) were implemented in Pascal and the programs executed on a HP workstation.

The graph of Figure 4.4 illustrates that the minimisation technique entitled DC, Full Gains was most effective in determining the optimum FPRM
expansions of incompletely specified Boolean functions. This indicates that an incompletely specified Boolean function should first be converted to the minium PPRM expansion. The optimum FPRM expansion can then be determined from this PPRM expression. It is interesting to note that both minimisation algorithms Full Gains, DC and DC, Full Gains performed significantly better than the Boolean matrix (DC) minimisation algorithm. The graph of Figure 4.5 illustrates that despite the differences in performance between the Boolean matrix (DC) and the remaining two minimisation algorithms the time taken by the algorithms to minimise groups of incompletely specified Boolean functions was not significantly different.

### 4.6 Summary

The minimal representation of any incompletely specified Boolean function may be realised through judicious use of the 'don't care' terms of the function. The work presented in this chapter has illustrated the operation of deriving optimal (sub-optimal) FPRM expansions from incompletely specified Boolean functions. Initially, the process of generating FPRM expansions from Boolean functions with 'don't care' terms was considered and existing methods for performing this task have been reviewed. A technique has been introduced which may be employed to derive the optimum allocation of 'don't care' terms of an incompletely specified Boolean function when this function is transformed to a polarity $p$ FPRM expansion ( $p$ is pre-determined). The technique does not perform an exhaustive search although the number of combination of values for the 'don't care' terms which must be evaluated is determined by the structure of the initial Boolean function. This method was incorporated into an algorithm and illustrated by an example using the tabular notation [1] and tabular method of generating FPRM expansions. The algorithm is not restricted to using this form of representation and may be employed with other notations e.g. RM coefficient maps, Boolean matrices. The use of this technique in conjunction with Harkings' technique [42] for generating the polarity matrix of any FPRM expansion is of particular interest and may prove to be efficient. However, this possibility has not yet been evaluated fully.

Finally, the determination of the optimum (sub-optimum) FPRM expansions of an incompletely specified Boolean function was discussed. Two heuristic algorithms have been presented where the order in which optimum (sub-


Figure 4.4: Percentage of randomly generated incompletely specified Boolean functions for which minimisation algorithms formed optimum FPRM expansions. (Boolean functions of 4-6 variables (1000 Boolean functions per set).)


Figure 4.5: Time taken for algorithms to optimise sets of 1000 incompletely specified Boolean functions
MCKenzie, L. M. $1995 \quad$ Chapter 4 106
optimum) polarity and optimum allocation of the 'don't care' terms is altered in each. The heuristic Full Gains minimisation algorithm of chapter 3 was employed to derive the polarity $p$ of the minimal (sub-minimal) FPRM expansions whilst the optimum use of the 'don't care' terms was determined by the technique described in section 4.3. Results have been presented which illustrate the performance of these algorithms and conclusions were drawn as to the significance of these results.

## Chapter 5

## Generating Kronecker Expansions Using a Tabular Technique

This chapter presents a tabular technique for representing and generating Kronecker expansions. The technique is an extended form of the tabular method of deriving fixed polarity Reed-Muller expansions from Boolean functions which was developed by Almaini, Thomson and Hanson [1].

The first section of this chapter details existing techniques for representing, deriving and optimising KRO expressions. Following this review the tabular technique for deriving KRO expansions is introduced. The technique may be employed to derive KRO expansion of both completely and incompletely specified Boolean functions. The full procedure for generating KRO expressions is fully detailed in section 5.3.
5.1 Review of Optimisation Techniques for Kronecker Expansions and Exclusive-OR Sum-of-Products Forms
The algorithms described in the following sections of this chapter may be employed to derive KRO expansions from Boolean functions. Hence, in this literature review alternative techniques for generating KRO expansions, other types of mixed polarity RM expansions and general ESOP forms are detailed. Additionally, in order to make the literature review as complete as possible, methods for deriving minimal ESOP forms are briefly described.

Bioul, Davio and Deschamps [64] described the $3^{n}$ canonical expressions (KRO expansions) which represent any $n$ variable Boolean function. These forms may be generated using transform matrices. The Boolean functions and KRO expansions are represented using Boolean matrices and the transform matrices constructed using the Kronecker product. The extended truth and weight vectors, which describe the coefficients and the weights of KRO expressions, respectively, were also introduced. The extended truth vector indicates the relationship between the $2^{n}$ coefficients of a $n$ variable RM expansion and the total of $3^{n}$ coefficients which may be used to
represent the expansion as a KRO expression. It is possible to determine the minimal KRO expansion representing an initial Boolean function by examining the extended weight vector. Additionally, Bioul et al determined minimal KRO expansions for Boolean functions of 3 and 4 variables. This was calculated by considering various classes of functions. Green [65] also considered the extended truth and weight vectors of KRO expansions and ternary maps were utilised to represent the extended truth vector. This type of map differs from a Karnaugh map and a RM coefficient map as it has a total of $3^{n}$ cells, where $n$ is the number of expansion variables. Techniques were introduced which may be employed to derive a ternary map of the extended truth vector from an initial ternary map representing the Boolean function or the RM expansion. The method is based on 'folding' the map and is similar to the manner in which RM coefficient maps may be derived from Karnaugh maps, as detailed in section 3.1 of chapter 3. Green also demonstrated that the ternary map representing an extended weight vector may be derived from the map representing the extended truth vector. Further, it is of interest to note that the techniques developed for deriving KRO expansion may be extended to realise PSDKRO expansions and quasi-Kronecker canonical forms. This was comprehensively demonstrated by Green [27]. Further research undertaken by Green and Khuwaja [66] considered KRO expansions represented by extended function vectors. Groups or cosets were formed where each coset comprised of all extended function vectors derived from a single truth vector. The relationships between the vectors of different cosets were identified and the value of this type of representation was explored with regard to determining minimal KRO expansions.

Lui and Muzio [52] identified fixed polarity modulo-2 canonical expansions and fixed basis modulo-2 canonical expansions which are identical to the FPRM and KRO expansions defined in this thesis. Algorithms which perform fast matrix transforms were presented and employed to efficiently derive fixed polarity and fixed basis expansions. Additionally, methods for exhaustively searching for minimal forms of both type of expansion were described. The techniques employ and generate expansions in Gray code and ternary Gray code sequences.

Fleisher, Tavel and Yeager [67] introduced the novel concept of the
exclusive-OR space. Here, KRO expansions adopt a graphical representation instead of the traditional algebraic form. This approach allows KRO expressions to be represented in a manner similar to that of using Boolean cubes to represent Boolean functions. Another type of graphical representation, the Kronecker Functional Decision Diagram, was presented by Sarabi, Ho, Irvani, Daasch and Perkowski [68]. These structures are similar to the BDDs described in section 2.6 of chapter 2. Indeed, ordered KFDDs include ordered BDDs [31] and Functional Decision Diagrams [48] as subsets. Sarabi et al described a technique for deriving minimal (subminimal) reduced ordered KFDD of Boolean functions. The technique is dependent on determining an optimal (sub-optimal) two-level KRO expression which can then be realised as a ROKFDD. The number of nodes in a ROKFDD is sensitive to the order of the variables in the structure, hence the optimisation problem becomes that of selecting a 'good' variable ordering. These structures are further considered in section 7.1 of chapter 7.

The different classes of ESOP forms were fully explored by Sasao [7]. Various functions were represented by each class of ESOP form and the numbers of product terms in each representation was presented. These results support the notions that, in general, the minimal form of any Boolean function will be contained within the broad class of ESOP forms, and that restricting minimisation algorithms to searching only subclasses will not result in optimal representations. Additionally, Sasao presented an algorithm for deriving PSDKRO expansions of Boolean functions.

Although many techniques exist for generating KRO expansions from Boolean functions there are, by comparison, relatively few minimisation methods. Techniques developed by Lui and Muzio [52] and Sarabi et al [68] have already been discussed. It is, therefore, of interest to consider briefly some methods for deriving minimal (sub-minimal) ESOP forms as these expressions include KRO expansions as a subclass. The development of exhaustive techniques for generating minimal ESOP expressions is prohibited by the complexity of this task. Any Boolean function may be represented by a large number of ESOP forms, as detailed in section 2.5.4 of chapter 2. As the number of variables in the Boolean function increases, the number of possible ESOP forms grows dramatically. This has necessitated the development of heuristic minimisation techniques which
endeavour to find optimal (sub-optimal) ESOP forms within a practical timescale. A number of these heuristic minimisation techniques are now reviewed.

The use of RM coefficient maps to represent and minimise FPRM expansions has previously been discussed. It is also possible to represent ESOP forms or mixed polarity expansions using Karnaugh maps and RM coefficient maps. Wu, Chen and Hurst [35] demonstrated that minimal (sub-minimal) ESOP forms may be derived by applying a similar minimisation strategy to the RM coefficient map as is applied to the Karnaugh map when minimising Boolean functions. Groups of terms are formed, with the dimensions $2^{k} \times 2^{m}(k, m \in$ $\{0,1, \ldots, n\}$ ), and the rules for minimisation require that cells containing 1 's be looped an odd number of times and, if necessary, cells containing 0 's be looped an even number of times. The product terms represented by these new groups may then be read from the map using a modified set of rules. It is also possible to apply the rules for grouping terms in reverse order and plot an ESOP expression on a map to represent a FPRM expansion. Tran [36] introduced a minimisation algorithm which operates on the RM coefficient map representing the FPRM expansion. The basic method is that of grouping terms according to the rules of $\mathrm{GF}(2)$ algebra. Additionally, the algorithm provides some means of determining which groups should be formed so as to realise a minimal (sub-minimal) ESOP form. Thus, the significance of the problem which is also inherent to Karnaugh map minimisation is addressed. The introduction of tri-state maps, suitable for representing ESOP forms is another extension to this graphical form of logic synthesis [38].

Habib [40] extended the original procedure for generating FPRM expansions from Boolean functions to include an algorithm which may be employed to generate minimal (sub-minimal) ESOP expressions. This is a heuristic technique which operates on the PPRM expansion represented in Boolean matrix form and uses 'minterm separation'. This operation partitions the matrix for each variable in turn making it possible to count the number of 'matching' coefficients between rows in the matrix. The variable with the highest count is the starting point, the Boolean matrix being partitioned accordingly. The EXOR operation which follows, takes into account each variable appearing in both true and complemented forms. Coefficients which
match in the matrix produce a single product term when the EXOR operation is performed, whilst terms which do not match will realise one or more terms in the new matrix. Hence maximum matching gives fewest product terms. The whole procedure is repeated until all variables have been appropriately transformed. Robinson and Yeh [69] presented a method for deriving a minimal ESOP expressions. The technique uses Boolean matrix representation and forms matrices for various ESOP expressions using the optimum FPRM expansion of the initial Boolean function as a starting point. The ESOP forms are constructed from the FPRM expansion and any one variable may be present in both true and complemented forms, all other variables are present in fixed polarity form. The method is extended to include expansions where more than one variable may be present in both true and complemented forms. The search for a minimum (sub-minimum) ESOP form involves the construction of a matrix comprised of the mixed polarity row vectors, then from this matrix deducing which combinations of rows reduce the number of product terms in the initial expansion.

Even, Kohavi and Paz [70] investigated the minimum number of product terms required to represent any switching function as an ESOP form. They specifically considered symmetric functions and stated upper bounds for the number of terms in FPRM expansions and ESOP forms. This work is particularly useful as it provides a means of determining the effectiveness of minimisation algorithms and techniques. Additionally, Even et al introduced a set of rules which may be employed to obtain an economic representation of a switching function as a FPRM expansion or ESOP expression. The rules of Merger, Exclusion, Increase of Order and Bridging were developed from modulo-2 algebra and are listed here in order of increasing complexity. The rules are applied successively, simplest first, and after each successful application the simplest rules are reapplied. This is repeated until the number of product terms in the ESOP form cannot be further reduced. Helliwell and Perkowski [71] extended the search for optimum mixed polarity RM expansions by introducing an operation called 'Xlinking'. This operation allows any two product terms to be expanded into an EXOR sum of one or more product terms which contain fewer literals. The product terms are not necessarily adjacent and do not need to contain the same number of literals. Two Xlinking operations are defined, the first, primary Xlinking relies on the substitution $x \oplus \bar{x}=1$ and operates on
terms of the same order which have the same literals present in opposite states. Although primary Xlinking may not reduce the number of product terms in an expression it will reduce the number of literals in these terms. Secondary Xlinking allows terms of different orders to be linked. The operation results in terms with fewer literals than the original lower order terms and another term of the same order as the higher order term. The Xlink algorithm utilises both primary and secondary Xlinking operations and simply tries to perform all Xlinks. The procedure does, however, have some order. All primary Xlinks should be carried out first, before moving on to secondary Xlinking and, if necessary, reiterating the procedure. Additionally, preference is given to performing simple Xlinks. The Xlinking rules can be applied to multiple output functions. In this case, all single output functions are minimised independently before being considered in conjunction with each other to determine which product terms should be Xlinked so as to optimise the multiple output function. It is also stated that the algorithm can be adapted to minimise incompletely specified functions. In addition to presenting a minimisation procedure for ESOP forms, Helliwell and Perkowski provided results indicating the performance of the algorithm when used to minimise a broad sample of Boolean functions. These results are valuable as they provide much needed information on the effectiveness of optimisation procedures and can act as a comparison for future work. Saul [72] further developed the rule-based method of obtaining a minimal (sub-minimal) ESOP representation. This work is based on the rules of Merger, Exclusion, Increase of Order and Bridging. Sub-algorithms are introduced which determine the best choice of product terms to link and the order in which the linking rules should be applied. The method of deciding which terms to link adopts a Quine-McCluskey type approach and determines all possible links. The technique for determining which rules should be applied provides a more efficient approach than that of the previous method. It categorises product terms depending on their order, thus product terms which obviously cannot be linked are not tested. Additionally, Saul proposed two new linking rules for multiple output functions, namely multiple output merging and multiple output bridging. These rules can be utilised to increase the efficiency of techniques for optimising multiple output functions. Incompletely specified Boolean functions are also considered. Only the merger rule is applied to these Boolean functions as it is considered that all other rules will increase the
number of product terms in the ESOP expression. The four rules defined previously [70] may be employed to form minimal (sub-minimal) ESOP forms from FPRM expansions or indeed initial ESOP expressions. However, one particular area where they do not produce optimum results is in failing to identify input irredundancy. This deficiency was highlighted by Pitty and Salmon [73] who proposed a linking rule, formed from the merger and bridging rules, as a solution. They suggested employing this rule in conjunction with the 4 rules developed by Even et al, thus improving the efficiency of the rule-based minimisation algorithms.

Fleisher, Tavel and Yeager [74] presented a heuristic minimisation algorithm based on the cube notation introduced in [67]. The technique links product terms of the initial RM expansion using three different operations. Two of these operations reduce the number of product terms in the expansion whilst the third is a restructuring operation. The principles which are employed in this technique are similar to those introduced by Even et al [70] and subsequently adopted by Helliwell and Perkowski [71].

Papakonstantinou [75] presented a technique for obtaining minimal ESOP forms from Boolean functions. The method involves the construction of a 'generation tree' formed from subfunctions of the original function and allocates weights as a means of determining optimum (sub-optimum) solutions. The algorithm is particularly suited to functions of 3 or 4 variables as it generates minimal ESOP forms. It can also be used with Boolean functions with larger numbers of variables provided sub-minimal solutions are acceptable.

Gatemap [76] is a logic synthesis system which employs RM minimisation techniques. The system generates three equations for each signal, two are Boolean expressions, the sum of products and the inverse sum of products forms. The third equation is the ESOP representation of the signal. The system proceeds to minimise these equations. The technique employed to optimise ESOP expressions is the rule-based method developed by Even et al [70]. The results of the minimisation algorithms are evaluated to obtain the most suitable circuit implementation. Gatemap appears to be the first synthesis system to utilise ESOP forms, others such as Socrates [77], employ Espresso-II [63] to minimise Boolean functions.

Programmable logic arrays (PLAs) are typically realised as an AND array and an OR array and are suitable for implementing Boolean functions. It has been suggested that a programmable device comprising of an AND array and an EXOR array and thus suited to RM applications could offer certain advantages as an alternative to the traditional form [5, 6, 16, 71]. These advantages include a reduction in the number of product terms required to implement a function and, hence, in the area of the device, and a device more suited to testing. The structure of PLAs with an EXOR array was investigated by Sasao [5] who determined the advantages of using different types of inputs to the AND array. The type of input provided affects the class of ESOP expression which could be implemented. A device without inverters or decoding logic at the input to the AND array can only realise a PPRM expansion, whilst providing input EXOR gates allows all FPRM expansions to be realised. It was, however, determined that the most beneficial implementation is a device with input decoders suitable for realising ESOP forms. Sasao also investigated the advantages of employing EXOR PLAs by establishing bounds for the number of product terms in the ESOP expressions of groups of Boolean functions. These bounds were calculated using three methods of minimising ESOP forms. The principles behind two of these methods are derived from the rules presented by Even et al [70]. Additionally, Sasao provided results for minimised functions which allow comparisons to be made between Boolean and RM implementations.

Sasao [6] extended his work on EXOR PLAs to develop a non-exhaustive minimisation algorithm. The technique is based on 7 rules and can be applied to multi-valued input two-valued output functions and is also suitable for multiple output functions. The algorithm forms a design method for EXOR PLAs with input decoders, hence, the minimisation procedure realises an ESOP expression. Additionally, Sasao provided results for the implementation of switching functions using AND-OR array devices and ANDEXOR array devices, both with 1 -bit and 2-bit input decoders. These results were generated by the algorithm previously outlined for generating ESOP forms and by exhaustively searching for the optimal Boolean representation.

A module generator which realises CMOS devices, XPLAs, which comprise of
an AND-plane and an EXOR-plane was developed by Froessl and Eschermann [16]. The EXOR-plane consists of a tree-like structure of interconnected 2input EXOR gates and it is possible to share EXOR gates common to several output functions. The XPLA and PLA implementations of logic functions were evaluated and comparisons show that using a XPLA-type device utilised less silicon area than PLA implementations for only a limited number of applications. The XPLA implementation was beneficial only when the number of product terms in the ESOP expression was significantly less than the number of product terms in the Boolean SOP representation. This can be attributed to the fact that the implementation of an EXOR gate requires a larger area than is necessary to realise AND, NAND and NOR gates.

Additional heuristic techniques for deriving minimal (sub-minimal) ESOP forms include the map-based method developed by Tran and Wang [78], which was also utilised as a part of an algorithm for minimising multiple output functions [79]. Green and Khuwaja [80] presented a heuristic technique based on principles similar to those employed in the QuineMcCluskey technique. This tabular approach searches for groups of 'adjacent terms' which may be combined to form a cover of the original expression. The rules of modulo-2 algebra are employed throughout. This type of approach was also adopted by Tran and Lee [81]. Riege and Besslich [61] presented the HEALEX system, a heuristic minimiser which generates optimal (sub-optimal) ESOP expressions from incompletely specified Boolean SOP forms. The strategy employed by this system is to derive a minimal (sub-minimal) FPRM expansion which can then be further minimised using heuristics to realise an ESOP form. As an alternative, Perkowski and Chrzanowska-Jeske [82] presented an intensive technique for deriving the absolute minimum ESOP form of any incompletely specified Boolean function.

### 5.2 Tabular Technique for Generating Kronecker Expansions

KRO expansions are a type of ESOP form with restrictions placed on the states of the expansion variables. These expansions have been described in section 2.5.1 of chapter 2 and their characteristics are now briefly summarised. Any Boolean function of $n$ variables may be represented by a total of $3^{n}$ KRO expansions. A KRO expression may be identified by observing the state of the expansion variables. That is, any variable which
appears in both true and complemented form throughout the expansion must appear once in each and every product term. This constraint is relaxed for variables which appear in either true or complemented form (but not both forms) throughout the expansion. Each canonical KRO expansion is identified by a polarity number $m\left(0 \leq m \leq 3^{n}-1\right)$ which indicates the state of each variable throughout the expansion.

It is also of interest to note that the polarity ( $3^{n}-1$ ) KRO expansion (all variables present in both true and complemented forms) representing a switching function is equivalent to the Boolean SOP representation (each product term is a minterm) when the inclusive-OR operator is replaced by the exclusive-OR operator. Finally, the $3^{n}$ KRO expansions include all $2^{n}$ FPRM expansions.

The tabular techniques reviewed in section 3.2 .1 of chapter 3 included a means of representing Boolean functions and FPRM expansions. This representation may be adapted in the following manner to allow the representation of KRO expansions. The tabular structure is once again adopted and each minterm of the Boolean function or product term of the KRO expansion is represented by the contents of a row of the table. Each column of the table represents a variable of the Boolean function or KRO expression. The table includes a header row which indicates the state of each variable.

Column $j(j=1,2, \ldots, n)$ should be headed
$x_{j}$ if literal $x_{j}$ is present throughout the expression.
$\bar{x}_{j}$ if literal $\bar{x}_{j}$ is present throughout the expression.
$\hat{x}_{j}$ if variable $x_{j}$ is present in both true and complemented forms throughout the expression.

Each cell of the table should contain either a 0 or a 1 , indicating the state of each variable in each minterm or product term. Let the binary $n$-tuple $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle$ denote a row of the table representing a Boolean function or KRO expression. Hence, any $c_{j}$ represents a cell of the table where $c_{j} \epsilon$ $\{0,1\}$ and $j=1,2, \ldots, n$. (It should be noted that $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle$ also denotes the condition of each variable in a minterm of a Boolean function or product term of a KRO expansion.) Consider a KRO expansion in which variable $\boldsymbol{x}_{\boldsymbol{j}}$
appears only in true (complemented) form throughout the expression. If literal $x_{j}\left(\bar{x}_{j}\right)$ is present in product term $\rho_{i}$ then column $j$ of the row representing $\rho_{i}$ should contain a 1 and this row is represented by the $n$ tuple $\left\langle c_{n} \ldots c_{j+1} 1 c_{j-1} \ldots c_{1}\right\rangle$. If, however, literal $x_{j}\left(\bar{x}_{j}\right)$ is absent from product term $\rho_{k}$ then column $j$ of the row representing $\rho_{k}$ should contain a 0 . This row is represented by the $n$-tuple $\left\langle c_{n} \ldots c_{j+1} 0 c_{j-1} \ldots c_{1}\right\rangle$. Now consider another KRO expansion where variable $x_{j}$ appears in both true and complemented forms. If literal $x_{j}\left(\bar{x}_{j}\right)$ is present in product term $\rho_{i}$ then column $j$ of the row representing $\rho_{i}$ should contain a $1(0)$ and the row is represented by the $n$-tuple $\left\langle c_{n} \ldots c_{j+1} 1 c_{j-1} \ldots c_{1}\right\rangle\left(\left\langle c_{n} \ldots c_{j+1} 0 c_{j-1} \ldots c_{1}\right\rangle\right)$. Note that variable $x_{j}$ must appear in every product term of the KRO expansion. If a Boolean function is being represented and literal $x_{j}\left(\bar{x}_{j}\right)$ is present in minterm $m_{i}$ then column $j$ of the row representing $m_{i}$ should contain a 1 ( 0 ). This row is represented by the $n$-tuple $\left\langle c_{n} \ldots c_{j+1} 1 c_{j-1} \ldots c_{1}\right\rangle\left(\left\langle c_{n} \ldots c_{j+1} 0 c_{j-1} \ldots c_{1}\right\rangle\right)(i, k$ $\in\left\{0,1, \ldots, 2^{n}-1\right\}$ ).

The following example illustrates the use of the tabular notation to represent a Boolean function and a KRO expansion.

Example 5.1 Display the following 3 variable Boolean function and KRO expansion using the tabular notation detailed previously.

Boolean function

$$
\left.\begin{array}{rl}
f\left(x_{3}, x_{2}, x_{1}\right) & =\bar{x}_{3} \bar{x}_{2} \bar{x}_{1}+\bar{x}_{3} x_{2} x_{1}+x_{3} \bar{x}_{2} \bar{x}_{1}+x_{3} x_{2} \bar{x}_{1} \\
\hat{x}_{3} & \hat{x}_{2} \\
\hline 0 & \hat{x}_{1} \\
\hline 0 & 0 \\
0 & 1 \\
1 & 0 \\
1 & 0 \\
1 & 1
\end{array}\right)
$$

KRO expansion (Polarity 21)

$$
f_{21}\left(x_{3}, x_{2}, x_{1}\right)=x_{3} \oplus x_{3} x_{1} \oplus \bar{x}_{3} x_{1} \oplus \bar{x}_{3} \bar{x}_{2}
$$

| $\hat{x}_{3}$ | $\bar{x}_{2}$ | $x_{1}$ |
| :--- | :--- | :--- |
| 1 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 0 | 1 |
| 0 | 1 | 0 |

(End of example)

The existing tabular technique of representing and generating FPRM expansions [1] may be used in two modes. The first mode operates by converting an initial Boolean function to the positive polarity RM expansion, then from this expression generating the FPRM expansion of the required polarity. In the second mode the initial Boolean function may be transformed to the necessary FPRM expansion using a modified form of this technique. It is also necessary to introduce an additional final step to 'adjust' the tabular representation of the FPRM expansion. Two new forms of the tabular technique are now described. The first method may be employed to derive a KRO expression from an initial Boolean function. The second technique is more general and may be used to derive a KRO expansion from either an initial Boolean function or from a KRO expansion of different polarity. The technique does not require the final 'adjustment' necessary in the tabular technique for deriving FPRM expressions [1].

The tabular notation described in the earlier part of this section is employed in the procedure to generate KRO expressions. However, an additional qualifier must be introduced. This will be called the bias of the variable and any variable may be said to be either positively or negatively biased. The bias of each variable in a Boolean function or KRO expression is indicated in the header row where the polarity of each variable is also shown. These column headings are now detailed.
If a variable $x_{j}$ is positively biased then column $j(j=1,2, \ldots, n)$ should be headed according to the notation displayed in the earlier part of this section (see also Example 5.1).
If variable $x_{j}$ is negatively biased then column $j(j=1,2, \ldots, n)$ should be headed
$\underline{x}_{j}$ if literal $\underline{x}_{j}$ is present throughout the expression.
$\underline{X}_{j}$ if literal $\overline{\underline{X}}_{j}$ is present throughout the expression.
$\hat{\underline{x}}_{j}$ if variable $\underline{x}_{j}$ is present in both true and complemented forms throughout the expression.
The bias of a variable does not affect the polarity of that variable. Instead, it indicates the significance of the $0 s$ and $1 s$ in the rows which represent each minterm or product term of the expression. If a variable is positively biased then the significance of these 0 s and 1 s is as described previously. The significance of these 0 s and 1 s for a negatively biased variable is now detailed. Once again each cell of the table should contain either a 0 or a 1 , indicating the state of each variable in each minterm or product term. The binary $n$-tuple $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle$ denotes a row of the table representing a Boolean function or KRO expression. Hence, any $c_{j}$ represents a cell of the table where $c_{j} \in\{0,1\}$ and $j=1,2, \ldots, n$. (The $n$-tuple $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle$ also denotes the condition of each variable in a minterm of a Boolean function or product term of a KRO expansion.) Consider a KRO expansion in which variable $x_{j}$ appears only in true (complemented) form throughout the expression. If literal $x_{j}\left(\bar{x}_{j}\right)$ is present in product term $\rho_{i}$ then column $j$ of the row representing $\rho_{i}$ should contain a 0 . This row is represented by the $n$-tuple $\left\langle c_{n} \ldots c_{j+1} 0 c_{j-1} \ldots c_{1}\right\rangle$. If, however, literal $x_{j}\left(\bar{x}_{j}\right)$ is absent from product term $\rho_{k}$ then column $j$ of the row representing $\rho_{k}$ should contain a 1 and the row is represented by the $n$-tuple $\left\langle c_{n} \ldots c_{j+1} 1 \quad c_{j-1} \ldots c_{1}\right\rangle$. Now consider another KRO expansion function where variable $x_{j}$ appears in both true and complemented forms. If literal $x_{j}\left(\bar{x}_{j}\right)$ is present in product term $\rho_{i}$ then column $j$ of the row representing $\rho_{i}$ should contain a 0 (1) and the row is represented by the $n$-tuple $\left.\left.<c_{n} \ldots c_{j+1} 0 c_{j-1} \ldots c_{1}\right\rangle\left(<c_{n} \ldots c_{j+1} 1 c_{j-1} \ldots c_{1}\right\rangle\right)$. Note that variable $x_{j}$ must appear in every product term of the KRO expansion. If a Boolean function is being represented then if literal $X_{j}\left(\bar{x}_{j}\right)$ is present in minterm $m_{i}$ then column $j$ of the row representing $m_{i}$ should contain a 0 (1). This row is represented by the $n$-tuple $\left\langle c_{n} \ldots c_{j+1} 0 c_{j-1} \ldots c_{1}\right\rangle$ $\left(\left\langle c_{n} \ldots c_{j+1} 1 c_{j-1} \ldots c_{1}\right\rangle\right)\left(i, k \in\left\{0,1, \ldots, 2^{n}-1\right\}\right)$.

The following examples illustrate the representation of a Boolean function and a KRO expansion using the modified tabular notation

Example 5.2 Display the following 3 variable Boolean function and KRO expansion using the tabular notation detailed previously. Variable $x_{3}$ is positively biased whilst variables $x_{2}$ and $x_{1}$ are negatively biased.

Boolean function

$$
\begin{aligned}
& f\left(x_{3}, x_{2}, x_{1}\right)=\bar{x}_{3} \bar{x}_{2} \bar{x}_{1}+\bar{x}_{3} x_{2} x_{1}+x_{3} \bar{x}_{2} \bar{x}_{1}+x_{3} x_{2} \bar{x}_{1} \\
& \quad \begin{array}{lll}
\hat{x}_{3} & \hat{\underline{x}}_{2} & \hat{\underline{x}}_{1} \\
\hline 0 & 1 & 1 \\
0 & 0 & 0 \\
1 & 1 & 1 \\
1 & 0 & 1
\end{array}
\end{aligned}
$$

KRO expansion (Polarity 21)

$$
\begin{aligned}
& f_{21}\left(x_{3}, x_{2}, x_{1}\right)=x_{3} \oplus x_{3} x_{1} \oplus \bar{x}_{3} x_{1} \oplus \bar{x}_{3} \bar{x}_{2} \\
& \begin{array}{lll}
\hat{x}_{3} & \underline{\underline{x}}_{2} & \underline{x}_{1} \\
\hline 1 & 1 & 1
\end{array} \\
& 1 \quad 10 \\
& 0 \quad 1 \quad 0 \\
& 0 \quad 0 \quad 1
\end{aligned}
$$

(End of example)

A table may be constructed to represent any Boolean function or KRO expansion by employing the notation which has been introduced. It is also necessary to perform the reverse operation, that is, to derive the KRO expansion from its tabular representation. This process is straightforward and is merely a reversal of the operation used to construct the table. It is, however, now briefly discussed to ensure both the clarity and integrity of this thesis. The polarity of each expansion variable throughout the KRO expression is indicated by the relevant column heading. The bias of each variable is also shown in the header row. The KRO expansion may be constructed by expanding each row of the table to form a single product term of the expression. The binary $n$-tuple $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle$ denotes a row and any $c_{j}$ represents a single cell and also a variable of the KRO expansion $\left(c_{j} \in\{0,1\}, j=1,2, \ldots, n\right)$.
Consider first positively biased variable $x_{j}$ and a row of the table represented by $\left\langle c_{n} \ldots c_{j+1} 0 c_{j-1} \ldots c_{1}\right\rangle$
if column $j$ is headed $x_{j}\left(\bar{x}_{j}\right)$ then literal $x_{j}\left(\bar{x}_{j}\right)$ is absent from the
product term $\boldsymbol{\rho}_{\boldsymbol{f}}$
if column $j$ is headed $\hat{x}_{j}$ then literal $\bar{x}_{j}$ is present in the product term $\rho_{i}$
Next consider positively biased variable $x_{j}$ and a row of the table represented by $\left\langle c_{n} \ldots c_{j+1} 1 c_{j-1} \ldots c_{1}\right\rangle$
if column $j$ is headed $x_{j}\left(\bar{x}_{j}\right)$ then literal $x_{j}\left(\bar{x}_{j}\right)$ is present in the product term $\boldsymbol{\rho}_{f}$
if column $j$ is headed $\hat{x}_{j}$ then literal $x_{j}$ is present in the product term $\rho_{1}$
Now variable $x_{j}$ is negatively blased and a row of the table is represented by $\left\langle c_{n} \ldots c_{j+1} 0 c_{j-1} \ldots c_{1}>\right.$
if column $j$ is headed $\underline{x}_{j}\left(\overline{\underline{x}}_{j}\right)$ then literal $x_{j}\left(\bar{x}_{j}\right)$ is present in the product term $\rho_{1}$
if column $j$ is headed $\hat{\underline{x}}_{j}$ then literal $x_{j}$ is present in the product term $\rho_{1}$
Finally, consider negatively biased variable $x_{j}$ and a row of the table represented by $\left\langle c_{n} \ldots c_{j+1} 1 c_{j-1} \ldots c_{1}\right\rangle$
if column $j$ is headed $x_{j}\left(\bar{x}_{j}\right)$ then literal $x_{j}\left(\bar{x}_{j}\right)$ is absent form the product term $\rho_{i}$
if column $j$ is headed $\hat{x}_{j}$ then literal $\bar{x}_{j}$ is present in the product term $\rho_{i}$
The product terms must be summed using modulo-2 addition (EXOR). If the final representation is the polarity ( $3^{n}-1$ ) KRO expansion (column headings are $\hat{x}_{j}, j=1,2, \ldots, n$ ) then this may be converted to a Boolean function simply by replacing the EXOR operator with the OR operator.

The use of this notation, which is more complex than that used in the original tabular technique, makes it possible to derive a KRO expansion from either a Boolean function or another KRO expression. The tabular technique for deriving a KRO expansion from an initial Boolean function is now described.

This tabular technique may be used to construct expansions comprised of mixed polarity variables and this procedure may be considered to consist of two steps. The steps are the generation of new product terms and deletion of equivalent terms. As the derivation of KRO expansions is an extension to the original tabular technique it is necessary to modify the
step of generating new product terms. The deletion of equivalent terms is unaffected. The step of generating new terms is now described. Consider first generating a polarity $m$ KRO expansion from a Boolean function. Consider $m_{j}$ of the ternary $n$-tuple $<m_{n} m_{n-1} \ldots m_{1}>$ where $m_{j}=0$, thus variable $x_{j}$ will appear in true form throughout the KRO expansion. Each variable of the Boolean function is positively biased. If a minterm of the Boolean function is represented by a row of the form $\left\langle c_{n} \ldots c_{j+1} 0 c_{j-1} \ldots c_{1}\right\rangle$ then a new row is generated which is represented by the $n$-tuple $\left\langle c_{n} \ldots c_{j+1} 1 c_{j-1} \ldots c_{1}\right\rangle$. This operation may be expressed algebraically,

$$
\dot{x}_{n} \dot{x}_{n-1} \ldots \bar{x}_{f} \ldots \dot{x}_{1}=\dot{x}_{n} \dot{x}_{n-1} \ldots x_{f} \ldots \dot{x}_{1} \oplus \dot{x}_{n} \dot{x}_{n-1} \ldots \dot{x}_{j+1} \dot{x}_{j-1} \ldots \dot{x}_{1}
$$

Hence, the row $\left\langle c_{n} \ldots c_{j+1} 0 c_{j-1} \ldots c_{1}\right\rangle$ now represents the product term $\dot{x}_{n} \dot{x}_{n-1}$ $\ldots \dot{x}_{j+1} \dot{x}_{j-1} \ldots \dot{x}_{1}$ whilst the new product term $\dot{x}_{n} \dot{x}_{n-1} \ldots x_{j} \ldots \dot{x}_{1}$ is represented by the $n$-tuple $\left\langle c_{n} \ldots c_{j+1} 1 c_{j-1} \ldots c_{1}>\right.$.

Now, consider generating a polarity $m$ KRO expansion from a Boolean function where $m_{j}$ of the ternary $n$-tuple $\left\langle m_{n} m_{n-1} \ldots m_{1}\right\rangle$ equals 1 , thus variable $x_{j}$ will appear in complemented form throughout the KRO expansion. Each variable of the Boolean function is positively biased. If a minterm of the Boolean function is represented by a row of the form $\left\langle c_{n} \ldots c_{j+1} 1 c_{j-1} \ldots c_{1}\right\rangle$ then a new row is generated which is represented by the n-tuple $<c_{n} \ldots c_{j+1} 0 c_{j-1} \ldots c_{1}>$. This operation may be expressed algebraically,

$$
\dot{x}_{n} \dot{x}_{n-1} \ldots x_{j} \ldots \dot{x}_{1}=\dot{x}_{n} \dot{x}_{n-1} \ldots \bar{x}_{f} \ldots \dot{x}_{1} \oplus \dot{x}_{n} \dot{x}_{n-1} \ldots \dot{x}_{j+1} \dot{x}_{j-1} \ldots \dot{x}_{1}
$$

Hence, the row $\left\langle c_{n} \ldots c_{j+1} 1 c_{j-1} \ldots c_{1}\right\rangle$ now represents the product term $\dot{x}_{n} \dot{x}_{n-1}$ $\ldots \dot{x}_{j+1} \dot{x}_{j-1} \ldots \dot{x}_{1}$ whilst the new product term $\dot{x}_{n} \dot{x}_{n-1} \ldots x_{j} \ldots \dot{x}_{1}$ is represented by the $n$-tuple $\left\langle c_{n} \ldots c_{j+1} 0 c_{j-1} \ldots c_{1}\right\rangle$. Variable $x_{j}$ is now negatively biased.

Finally, consider generating a polarity $m$ KRO expansion from a Boolean function where $m_{j}$ of the ternary $n$-tuple $\left\langle m_{n} m_{n-1} \ldots m_{1}\right\rangle$ equals 2 , thus variable $x_{j}$ will appear in both true and complemented forms throughout the KRO expansion. Each variable of the Boolean function is positively biased. Each minterm of the Boolean function contains variable $x_{f}$ in either true or complemented form. As this is the representation required in the polarity $m$ KRO expansion then no new terms need be generated for variable $x_{j}$.

The following algorithm may be employed to derive a polarity $m$ KRO expansion from a Boolean function.

As defined previously, the binary $n$-tuple $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle$ denotes a row of the
table representing a Boolean function or KRO expression. Hence, any $c_{j}$ represents a cell of the table where $c_{j} \in\{0,1\}$ and $j=1,2, \ldots, n .\left(\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle\right.$ also denotes the condition of each variable in a minterm of a Boolean function or product term of a KRO expansion.)
The ternary $n$-tuple $\left.m\left(<m_{n} m_{n-1} \ldots m_{1}\right\rangle\right)$ indicates the polarity of a KRO expansion ( $m_{j} \in\{0,1,2\}$ ).

## Converting a Boolean function to a polarity m KRO expansion

S. 1 Represent the $n$ variable Boolean function using the tabular notation. Thus for $j=1, \ldots, n$ form a column with the heading $\hat{x}_{j}$, i.e. each function variable is present in both true and complemented forms and each variable is positively biased. Form the rows of the table where each row represents a minterm. Each cell of the table is filled with 0 or 1 according to the rules defined previously. A binary $n$ tuple $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle$ denotes each row of the table. Let $h=n$.
S. 2 Let $j=h$.

If $m_{j}=0\left(m_{j}=1\right)$ then
if any row of the table (minterm of the Boolean function or product term of the KRO expansion) has a representation of the form $\left\langle c_{n} \ldots c_{j+1} 0 c_{j-1} \ldots c_{1}\right\rangle\left(\left\langle c_{n} \ldots c_{j+1} 1 c_{j-1} \ldots c_{1}\right\rangle\right)$ then generate a new row which is represented by the $n$-tuple $<c_{n} \ldots c_{j+1} 1 c_{j-1}$ $\left.\ldots c_{1}\right\rangle\left(\left\langle c_{n} \ldots c_{j+1} 0 c_{j-1} \ldots c_{1}\right\rangle\right)$. If this new row is identical to any row already existing in the table then delete both the existing row and the new row. Otherwise add the new row to the foot of the table.

If $m_{j}=2$ then do not generate any new terms.
Repeat this step until all rows of the table have been evaluated.
S. 3 If $m_{J}=0$ then
alter column heading according to the polarity of the variable. New column heading is $\boldsymbol{x}_{\boldsymbol{f}}$
If $m_{j}=1$ then
alter column heading according to the polarity of the variable and change the bias of the variable. New column heading is $\underline{\bar{x}}_{j}$ If $m_{j}=2$ then column heading does not change.
S. 4 If $h>1$ then let $h=h-1$ and go to S.2. Otherwise the table represents the polarity $m$ KRO expansion. The state of each variable is indicated by the header row of the table.

The polarity $m$ KRO expansion may be constructed from the final table generated by the algorithm which has just been detailed. The rules for interpreting this tabular representation have already been developed. This algorithm was implemented in Pascal and the programs executed on a Dell P60 personal computer. The following example illustrates the generation of a KRO expansion from a Boolean function using the algorithm previously detailed.

Example 5.3 Derive the polarity 69 KRO expansion of the 4 variable Boolean function

$$
\begin{aligned}
f\left(x_{4}, x_{3}, x_{2}, x_{1}\right)= & \sum m(3,4,5,6,8,9,13,14,15) \\
= & \bar{x}_{4} \bar{x}_{3} x_{2} x_{1}+\bar{x}_{4} x_{3} \bar{x}_{2} \bar{x}_{1}+\bar{x}_{4} x_{3} \bar{x}_{2} x_{1}+\bar{x}_{4} x_{3} x_{2} \bar{x}_{1}+x_{4} \bar{x}_{3} \bar{x}_{2} \bar{x}_{1} \\
& +x_{4} \bar{x}_{3} \bar{x}_{2} x_{1}+x_{4} x_{3} \bar{x}_{2} x_{1}+x_{4} x_{3} x_{2} \bar{x}_{1}+x_{4} x_{3} x_{2} x_{1}
\end{aligned}
$$

Transform the Boolean function to the polarity 69 KRO expansion, applying the transformation rules.
S. 1 Represent the 4 variable Boolean function using the tabular notation. S. 2 - S. $4 \quad m_{4}=2$, therefore do not generate any new terms. Column heading does not change. $h=4$ therefore go to S.2.
S. $2 m_{3}=1$, therefore generate new product terms.
( ${ }_{i}$ denotes equivalent terms)
Generated terms


Cancel all equivalent product terms and add any remaining new product terms to the foot of the table.
S. 3 Alter the column heading to indicate the polarity of the variable ( $\bar{x}_{3}$ ) and change the bias of the variable.
The table represents the polarity 71 KRO expansion.
S. $4 \quad h=3$ therefore go to S.2.
S. 2 - S. $4 m_{2}=2$, therefore do not generate any new terms. Column heading does not change. $h=2$ therefore go to S.2.
S. $2 m_{1}=0$, therefore generate new product terms.

Generated terms
Polarity 71 KRO expansion

${ }^{*}$|  | $\hat{x}_{4}$ | $\underline{\underline{x}}_{3}$ | $\hat{x}_{2}$ | $\hat{x}_{1}$ |
| :--- | :--- | :--- | :--- | :--- |
|  | 0 | 0 | 1 | 1 |
|  | 0 | 1 | 0 | 0 |

$\begin{array}{lllll}* & 0 & 1 & 0 & 1\end{array}$
$\begin{array}{llll}0 & 1 & 1 & 0\end{array}$
1000
$\begin{array}{llll}1 & 1 & 0 & 1\end{array}$
$\begin{array}{llll}1 & 1 & 1 & 0\end{array}$
$\begin{array}{lllll}* & 1 & 1 & 1 & 1\end{array}$
$0 \quad 0 \quad 0 \quad 0$
$\begin{array}{lllll}{ }_{5} & 0 & 0 & 0 & 1\end{array}$
$0 \quad 0 \quad 1 \quad 0$
10010
${ }^{*}{ }_{6} \quad 1 \quad 0 \quad 1 \quad 1 \quad 1$

Cancel all equivalent product terms and add any remaining new product terms to the foot of the table.
S. 3 Column heading to indicate the polarity of the variable $\left(x_{1}\right)$ and change the bias of the variable.
The table represents the polarity 69 KRO expansion.
S. $4 \quad h=1$. The Boolean function has been converted to the polarity 69 KRO expansion.

Polarity 69 KRO expansion

| $\hat{x}_{4}$ | $\overline{\underline{x}}_{3}$ | $\hat{x}_{2}$ | $x_{1}$ |  |
| :--- | :--- | :--- | :--- | :--- |
| 0 | 1 | 0 | 0 | $\left(\bar{x}_{4} \bar{x}_{2}\right)$ |
| 0 | 1 | 1 | 0 | $\left(\bar{x}_{4} x_{2}\right)$ |
| 1 | 0 | 0 | 0 | $\left(x_{4} \bar{x}_{3} \bar{x}_{2}\right)$ |
| 1 | 1 | 0 | 1 | $\left(x_{4} \bar{x}_{2} x_{1}\right)$ |
| 1 | 1 | 1 | 0 | $\left(x_{4} x_{2}\right)$ |
| 0 | 0 | 0 | 0 | $\left(\bar{x}_{4} \bar{x}_{3} \bar{x}_{2}\right)$ |
| 0 | 0 | 1 | 0 | $\left(\bar{x}_{4} \bar{x}_{3} x_{2}\right)$ |
| 1 | 0 | 1 | 0 | $\left(x_{4} \bar{x}_{3} x_{2}\right)$ |
| 0 | 1 | 1 | 1 | $\left(\bar{x}_{4} x_{2} x_{1}\right)$ |
| 1 | 0 | 0 | 1 | $\left(x_{4} \bar{x}_{3} \bar{x}_{2} x_{1}\right)$ |

The Boolean function has been transformed to the equivalent polarity 69 KRO expansion and is represented by the following equation.

$$
\begin{aligned}
f_{\theta 9}\left(x_{4}, x_{3}, x_{2}, x_{1}\right)= & x_{4} x_{2} \oplus x_{4} \bar{x}_{2} x_{1} \oplus x_{4} \bar{x}_{3} x_{2} \oplus x_{4} \bar{x}_{3} \bar{x}_{2} \oplus x_{4} \bar{x}_{3} \bar{x}_{2} x_{1} \\
& \oplus \bar{x}_{4} x_{2} \oplus \bar{x}_{4} x_{2} x_{1} \oplus \bar{x}_{4} \bar{x}_{2} \oplus \bar{x}_{4} \bar{x}_{3} x_{2} \oplus \bar{x}_{4} \bar{x}_{3} \bar{x}_{2}
\end{aligned}
$$

(End of example)

The preceding algorithm has detailed a technique for deriving a polarity $m$ KRO representation from an initial Boolean function. It may, however, be necessary to derive a polarity $m$ KRO expansion from another KRO expansion of different polarity. A more general technique is now detailed in which any Boolean function or polarity $r$ KRO expansion may be converted to a polarity $m$ KRO expansion. The step of generating new terms must be modified. When generating terms the current polarity and new polarity of each variable (as indicated by $r$ and $m$, respectively), must be taken into account together with the bias of that variable. The following algorithm forms a polarity $m$ KRO expansion from an initial expression which may be either a Boolean function or polarity $r$ KRO form.

The binary $n$-tuple $\left\langle c_{n} \ldots c_{1}\right\rangle$ is as defined previously and denotes a row of
the table representing a Boolean function or KRO expression. Hence, any $c_{j}$ represents a cell of the table where $c_{j} \in\{0,1\}$ and $j=1,2, \ldots, n$.
The ternary $n$-tuples $m\left(\left\langle m_{n} m_{n-1} \ldots m_{1}\right\rangle\right)$ and $r\left(\left\langle r_{n} r_{n-1} \ldots r_{1}\right\rangle\right)$ indicate the polarity of KRO expansions $\left(m_{j}, r_{j} \in\{0,1,2\}\right.$ ).

## Generating a polarity m KRO expansion

S. 1 Construct a table representing the $n$ variable Boolean function or KRO expansion using the tabular notation detailed previously. For $j$ $=1,2, \ldots, n$ form a column with a heading which indicates both the polarity of the variable $x_{j}$ and its bias. Form the rows of the table where each row represents a minterm or product term. Each cell of the table is filled with a 0 or a 1 according to the rules defined previously. A binary $n$-tuple $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle$ denotes each row of the table. Let $h=n$.
Let $r$ be the polarity of this initial expansion. If the initial expression is a Boolean function then let $r=3^{n}-1$.
S. 2 Let $j=h$.

If $m_{j}=r_{j}$ then no change in the polarity of variable $x_{j}$, go to S.5. Otherwise, go to S.3.
S. 3 If variable $x_{j}$ is positively biased then determine whether $r_{j}$ and $m_{j}$ satisfy any one of the following conditions.
a) $\quad r_{j}=0, m_{j}=1$
b) $\quad r_{j}=1, m_{j}=0$
c) $\quad r_{j}=2, m_{j}=1$
d) $\quad r_{j}=2, m_{j}=0$

If $r_{j}$ and $m_{j}$ satisfy any one of the conditions a - $c(d)$ then if any row of the table (minterm of the Boolean function or product term of the KRO expansion) has a representation of the form $\left\langle c_{n} \ldots c_{j+1} 1 c_{j-1} \ldots c_{1}\right\rangle\left(\left\langle c_{n} \ldots c_{j+1} 0 c_{j-1} \ldots c_{1}\right\rangle\right)$ then generate a new row which is represented by the $n$-tuple $<c_{n} \ldots c_{j+1} 0 c_{j-1}$ $\left.\ldots c_{1}\right\rangle\left(\left\langle c_{n} \ldots c_{j+1} 1 c_{j-1} \ldots c_{1}\right\rangle\right)$.
If variable $x_{j}$ is negatively biased then determine whether $r_{j}$ and $m_{j}$ satisfy any one of the following conditions.
e) $\quad r_{j}=0, m_{j}=1$
f) $\quad r_{j}=1, m_{j}=0$
g) $\quad r_{j}=2, m_{j}=1$
h) $\quad r_{j}=2, m_{j}=0$

If $r_{j}$ and $m_{j}$ satisfy any one of the conditions $\mathbf{e}-\mathbf{g}(\mathbf{h})$ then if any row of the table (minterm of the Boolean function or product term of the KRO expansion) has a representation of the form $\left\langle c_{n} \ldots c_{j+1} 0 c_{j-1} \ldots c_{1}\right\rangle\left(\left\langle c_{n} \ldots c_{j+1} 1 c_{j-1} \ldots c_{1}\right\rangle\right)$ then generate a new row which is represented by the $n$-tuple $<c_{n} \ldots c_{j+1} 1 c_{j-1}$ $\left.\ldots c_{1}\right\rangle\left(\left\langle c_{n} \ldots c_{j+1} 0 c_{j-1} \ldots c_{1}\right\rangle\right)$.
If $m_{j}=2$ and $x_{j}$ is positively (negatively) biased then
if any row of the table (minterm of the Boolean function or product term of the KRO expansion) has a representation of the form $\left\langle c_{n} \ldots c_{j+1} 0 c_{j-1} \ldots c_{1}\right\rangle\left(\left\langle c_{n} \ldots c_{j+1} 1 c_{j-1} \ldots c_{1}\right\rangle\right)$ then generate a new row which is represented by the $n$-tuple $\left\langle c_{n} \ldots c_{j+1} 1 c_{j-1}\right.$ $\left.\ldots c_{1}\right\rangle\left(\left\langle c_{n} \ldots c_{j+1} 0 c_{j-1} \ldots c_{1}\right\rangle\right)$.
If this newly generated row is identical to any row already existing in the table then delete both the existing row and the new row. Otherwise add the new row to the foot of the table. Repeat this step until all rows have been evaluated.
S. 4 If $r_{j}=0$ and $m_{j}=1$ or 2 then
alter column heading according to the polarity of the variable $x_{j}$ The bias of the variable is unchanged.
If $r_{j}=1$ or 2 and $m_{j}=0$ then
alter column heading according to the polarity of the variable $x_{j}$ The bias of the variable is unchanged.
If $r_{j}=1(2)$ and $m_{j}=2(1)$ then
alter column heading according to the polarity of the variable $x_{j}$. The bias of the variable is reversed i.e. a positively biased variable becomes negatively biased and vice versa.
S. 5 If $h>1$ then let $h=h-1$ and go to S.2. Otherwise the table represents the polarity $m$ KRO expansion. The state of each variable is indicated by the header row of the table.
The polarity $m$ KRO expansion may be constructed from the final table generated by the preceding algorithm. The rules for interpreting the tabular representation have already been developed.

The above algorithm was implemented in Pascal and the programs executed on a Dell P60 personal computer.

The following example illustrates the conversion of a polarity 69 KRO
expansion to a polarity 76 KRO expansion.

Example 5.4 Derive the polarity 76 KRO expansion from the equivalent polarity 69 KRO expression.

$$
\begin{aligned}
f_{\epsilon \theta}\left(x_{4}, x_{3}, x_{2}, x_{1}\right)= & x_{4} x_{2} \oplus x_{4} \bar{x}_{2} x_{1} \oplus x_{4} \bar{x}_{3} x_{2} \oplus x_{4} \bar{x}_{3} \bar{x}_{2} \oplus x_{4} \bar{x}_{3} \bar{x}_{2} x_{1} \\
& \oplus \bar{x}_{4} x_{2} \oplus \bar{x}_{4} x_{2} x_{1} \oplus \bar{x}_{4} \bar{x}_{2} \oplus \bar{x}_{4} \bar{x}_{3} x_{2} \oplus \bar{x}_{4} \bar{x}_{3} \bar{x}_{2}
\end{aligned}
$$

Transform the polarity 69 KRO expansion to the polarity 76 KRO expression, applying the transformation rules.
S. 1 Represent the 4 variable polarity 69 KRO expansion using the tabular notation. $r=69, m=76$. (Polarity 69 KRO expansion was generated in Example 5.3.)
S. 2 - S. $5 \quad m_{4}=r_{4}=2$, therefore do not generate any new terms. Column heading does not change. $h=4$ therefore go to S.2.
S. $2 m_{3}=2, r_{3}=1$ therefore $m_{3} \neq r_{3}$.
S. 3 Generate new product terms.
( ${ }_{i}$ denotes equivalent terms)
Generated terms


Cancel all equivalent product terms and add any remaining new
product terms to the foot of the table.
S. 4 Alter the column heading to indicate the polarity of the variable ( $\hat{X}_{3}$ ) and change the bias of the variable.
The table represents the polarity 78 KRO expansion.
S. $5 \quad h=3$ therefore go to S.2.
S. $2 m_{2}=1, r_{2}=2$ therefore $m_{2} \neq r_{2}$.
S. 3 Generate new product terms.

Generated terms
Polarity 78 KRO expansion

$$
\left(\underline{\bar{x}}_{2}\right)
$$

| $\hat{x}_{4}$ | $\hat{X}_{3}$ | $\hat{X}_{2}$ | $x_{1}$ |
| :--- | :--- | :--- | :--- |
| 1 | 1 | 1 | 0 |


${ }^{*}$|  | $\hat{x}_{4}$ | $\hat{x}_{3}$ | $\underline{\underline{x}}_{2}$ |
| :---: | :---: | :---: | :---: |
|  | $x_{1}$ |  |  |
| 1 | 1 | 0 | 0 |
|  | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
|  | 0 | 0 | 0 |
|  |  | 1 |  |

$\begin{array}{llll}0 & 1 & 1 & 1\end{array}$
${ }^{*} \begin{array}{lllll}5 & 0 & 1 & 0 & 0\end{array}$
$\begin{array}{llll}0 & 0 & 1 & 1\end{array}$
Cancel all equivalent product terms and add any remaining new product terms to the foot of the table.
S. 4 Alter the column heading to indicate the polarity of the variable ( $\bar{x}_{2}$ ) and change the bias of the variable.
The table represents the polarity 75 KRO expansion.
S. $5 \quad h=2$ therefore go to S.2.
S. $2 m_{1}=1, r_{1}=0$ therefore $m_{1} \neq r_{1}$.
S. 3 Generate new product terms.

Generated terms
Polarity 75 KRO expansion
$\left(\bar{x}_{1}\right)$


Cancel all equivalent product terms and add any remaining new product terms to the foot of the table.
S. 4 Alter the column heading to indicate the polarity of the variable ( $\bar{x}_{1}$ ) and change the bias of the variable.
The table represents the polarity 76 KRO expansion.
S. $5 h=1$. The polarity 69 KRO expansion has been converted to the polarity 76 KRO expansion.
Polarity 76 KRO expansion

| $\hat{x}_{4}$ | $\hat{\underline{x}}_{3}$ | $\overline{\underline{x}}_{2}$ | $\bar{x}_{1}$ |  |
| :--- | :--- | :--- | :--- | :--- |
| 1 | 1 | 1 | 0 | $\left(x_{4} x_{3}\right)$ |
| 1 | 1 | 0 | 1 | $\left(x_{4} x_{3} \bar{x}_{2} \bar{x}_{1}\right)$ |
| 1 | 0 | 0 | 0 | $\left(x_{4} \bar{x}_{3} \bar{x}_{2}\right)$ |
| 0 | 1 | 1 | 1 | $\left(\bar{x}_{4} x_{3} \bar{x}_{1}\right)$ |
| 0 | 0 | 1 | 1 | $\left(\bar{x}_{4} \bar{x}_{3} \bar{x}_{1}\right)$ |
| 0 | 1 | 0 | 1 | $\left(\bar{x}_{4} x_{3} \bar{x}_{2} \bar{x}_{1}\right)$ |
| 0 | 0 | 0 | 1 | $\left(\bar{x}_{4} \bar{x}_{3} \bar{x}_{2} \bar{x}_{1}\right)$ |
| 0 | 0 | 1 | 0 | $\left(\bar{x}_{4} \bar{x}_{3}\right)$ |
| 0 | 1 | 0 | 0 | $\left(\bar{x}_{4} x_{3} \bar{x}_{2}\right)$ |
| 0 | 0 | 0 | 0 | $\left(\bar{x}_{4} \bar{x}_{3} \bar{x}_{2}\right)$ |

The polarity 69 KRO expansion has been transformed to the equivalent polarity 76 KRO expansion and is represented by the following equation.

$$
\begin{aligned}
f_{76}\left(x_{4}, x_{3}, x_{2}, x_{1}\right)= & x_{4} x_{3} \oplus x_{4} x_{3} \bar{x}_{2} \bar{x}_{1} \oplus x_{4} \bar{x}_{3} \bar{x}_{2} \oplus \bar{x}_{4} x_{3} \bar{x}_{1} \oplus \bar{x}_{4} x_{3} \bar{x}_{2} \\
& \oplus \bar{x}_{4} x_{3} \bar{x}_{2} \bar{x}_{1} \oplus \bar{x}_{4} \bar{x}_{3} \oplus \bar{x}_{4} \bar{x}_{3} \bar{x}_{1} \oplus \bar{x}_{4} \bar{x}_{3} \bar{x}_{2} \oplus \bar{x}_{4} \bar{x}_{3} \bar{x}_{2} \bar{x}_{1}
\end{aligned}
$$

## (End of example)

5.3 Tabular Technique for Generating Kronecker Expansions from Incompletely Specified Boolean Functions
Kronecker (KRO) expansions are a subset of all the ESOP forms which may be used to represent any switching function. This group, comprising of $3^{n}$ expansions, includes all $2^{n}$ FPRM expressions. The tabular techniques detailed in the preceding section may be employed to derive a KRO expansion from a Boolean function or another KRO expression. However, the techniques are suitable only for completely specified Boolean functions. A technique, described in section 4.3 of chapter 4 , may be used to construct polarity $p$ FPRM expansions from an incompletely specified Boolean function, where $p$ is predetermined. The technique detects the allocation of 'don't care' terms which maximally reduces the number of product terms in the FPRM expansion. It is possible to employ this technique in conjunction with either of the algorithms detailed in the preceding section to derive polarity $m$ KRO expansions from incompletely specified Boolean functions. The polarity, $m$, must be predetermined and the expressions are the minimal polarity $m$ KRO expansions of the incompletely specified Boolean function. The technique is now detailed.

Converting an incompletely specified Boolean function to polarity m KRO expansion.
Employ the algorithm detailed in section 4.3 of chapter 4, where each reference to polarity $p$ and a polarity $p$ FPRM expansion is instead a reference to polarity $m$ and a polarity $m$ KRO expansion. S. 2 of the algorithm is replaced by $\mathrm{S} .2^{\prime}$.
S.2' Transform the Boolean function the polarity $m$ KRO expansion, using either of the algorithms detailed in section. Transform the 'don't care' terms according to the rules detailed in section 4.4 of chapter 4.

The final table represents the minimal $m$ KRO expansion derived from incompletely specified Boolean function and may be interpreted according to the definitions detailed in the previous section.

### 5.4 Summary

Any Boolean function may be represented by a total of $3^{n}$ KRO expansions. These expressions include all $2^{n}$ FPRM expansion where expansion variables appear in either true or complemented form. The remaining KRO expansions may be termed mixed polarity forms as any expansion variable may appear in both true and complemented form throughout the expression. A variable which is present in both states must appear in each and every product term in the expression. Thus, KRO expressions constitute only a small, clearly defined subset of all possible ESOP forms of any switching function. Simple tabular methods of deriving KRO expansions have been described in the preceding sections of this chapter. These techniques may be employed to generate KRO expression from Boolean functions or from initial KRO expansions of different polarity. Additionally, the KRO expression representing any incompletely specified Boolean function may be generated. The optimal allocation of the 'don't care' terms is determined using the technique detailed in chapter 4. This work is, therefore, a valuable extension to the tabular technique developed by Almaini et al [1]. Simple tools have been developed which may be employed in an exhaustive search for minimal KRO expansions. This is not an insignificant task and perhaps the most efficient method of generating one KRO expression from an existing KRO expansion is to use ternary Gray code ordering. Additionally, the optimum allocation of any 'don't care' terms may be determined. Thus, it is possible to realise the minimum (sub-minimum) KRO expansions of an incompletely specified Boolean function, where only the polarity of the KRO expressions must be determined through exhaustive search.

## Chapter 6

## Generating Kronecker Expansions from Reduced Boolean Sum-of- <br> Products Forms

This chapter presents a technique which may be employed to generate Kronecker expansions from reduced Boolean sum-of-products forms. A tabular technique, developed by Almaini, Thomsom and Hanson [1] and reviewed in chapter 3, generates FPRM expansions from Boolean functions where each product term of the function is a minterm. If, however, the initial representation is a Boolean SOP form where some, or indeed all, product terms are not minterms then an additional transformation must be performed before the conversion to a FPRM expansion may be initiated. This conversion involves expanding each product term to minterm form and, if the Boolean SOP representation is not disjoint, then duplicate minterms must be removed. This operation may result in a significant increase in the number of terms in the Boolean function as well as introducing an extra step into the transformation procedure.

The following section reviews techniques for deriving exclusive-OR sum-ofproducts expressions from reduced Boolean SOP forms. An extended form of the tabular technique is then presented. This new technique provides a means of generating KRO expansions from reduced Boolean SOP forms which are comprised of disjoint product terms. The KRO expansions are generated directly from the disjoint product terms hence the Boolean SOP expression need not be expanded to its canonical form.

### 6.1 Review of Techniques for Deriving Exclusive-OR Sum-of-Products

 FormsA variety of techniques for deriving FPRM and KRO expansions of Boolean functions have been reviewed in chapters 3 and 5. The techniques operate on an initial Boolean function where each product term of the expression is a minterm. The methods now detailed may be employed to generate FPRM
expansions and ESOP forms from Boolean SOP forms where the product terms of each Boolean expression are disjoint. That is, the Boolean function is expressed in canonical form.

Fisher [83] introduced a technique for deriving FPRM expansions from disjoint Boolean SOP forms. The technique constructs a $2^{n} x 2^{n}$ matrix where each row of the matrix represents a polarity $p$ FPRM expansion of the initial Boolean SOP form ( $p=0,1, \ldots, 2^{n}-1$ ). Each column of the matrix denotes a single $b_{i}$ coefficient of a FPRM expansion ( $i=0,1, \ldots, 2^{n}-1$ ). It is necessary to construct a matrix for each product term of the initial Boolean SOP form then the matrices should be summed using modulo-2 addition. The resulting matrix represents all $2^{n}$ FPRM expansions of the initial disjoint Boolean SOP form.

An alternative approach is detailed by Purwar [47] who derived FPRM expansions from Boolean functions represented by BDDs. This technique exploits a particular feature of BDDs, namely, that the Boolean function is always represented as a disjoint SOP form. Any path through a BDD which terminates in a node with the value 1 will contribute a product term to the equivalent FPRM expansion. Paths which terminate in nodes with value 0 need not be evaluated. The value of each $b_{i}$ coefficient of any FPRM expansion is determined by evaluating the numbers of minterms represented on each of the relevant paths of the BDD.

Falkowski and Perkowski [60] described a technique where each product term of the initial disjoint SOP form is expanded to represent the equivalent product terms of the polarity $p$ FPRM expansion. Duplicate product terms must then be located and deleted before the final FPRM expansion is realised. The method described by Sarabi and Perkowski [46] introduces the operations of cube commonality, difference and symmetric difference. These algebraic operations are performed on the product terms of the initial disjoint Boolean SOP form in order to generate equivalent FPRM expansions. Additionally, Varma and Trachtenberg [62] introduced a means of deriving a AND-EXOR covers (ESOP forms) from initial Boolean SOP forms and incorporated this technique with their method for transforming incompletely specified Boolean functions to PPRM expansions.
6.2 Tabular Techniques for Generating Kronecker Expansions from Reduced Boolean Sum-of-Products Forms
The work presented in this section includes details of an algorithm which may be employed to derive KRO expansions from reduced Boolean SOP forms. It is necessary that the initial Boolean representation on which the algorithm operates is comprised of disjoint product terms. Hence, this section commences with a review of the relationships between Boolean functions and ESOP forms. Additionally, the construction of disjoint SOP forms is discussed.

### 6.2.1 Reduced Boolean Sum-of-Products Forms

Any $n$ variable switching function may be represented in Boolean SOP form

$$
\begin{align*}
f\left(x_{n}, x_{n-1} \cdots, \ldots, x_{1}\right) & =\sum_{i=0}^{2 n-1} d_{i} m_{i} \\
& =d_{0} \bar{x}_{n} \bar{x}_{n-1} \ldots \bar{x}_{2} \bar{x}_{1}+d_{1} \bar{x}_{n} \bar{x}_{n-1} \ldots \bar{x}_{2} x_{1}+d_{2} \bar{x}_{n} \bar{x}_{n-1} \ldots x_{2} \bar{x}_{1}+\ldots \ldots+d_{2^{n-1}} x_{n} x_{n-1} \ldots x_{2} x_{1} \tag{5.1}
\end{align*}
$$

where $m_{i}$ denotes a minterm of the function, $d_{i} \in\{0,1\}$ is an operational domain coefficient and $x_{j}$ and $\bar{x}_{j}$ are literals of the function, in true and complemented forms respectively $\left(i=0,1, \ldots, 2^{n}-1, j=1,2, \ldots, n\right)$. This expression is the canonical disjunctive form. A minterm $m_{i}$ is defined as a product of function variables and each minterm comprises of every function variable in either true or complemented form. The canonical Boolean SOP form may, however, be an inefficient representation of a switching function. A more economical form may be realised by employing the rules of Boolean algebra to combine minterms to form product terms. The minimisation of Boolean functions has been extensively studied and will not be reviewed here as it is beyond the scope of this research project. The techniques presented in this chapter may be used to derive KRO expansions from minimised Boolean functions. A Boolean function which is not a canonical form, that is each term is not a minterm, will henceforth be described as a reduced Boolean SOP form. Any term of a reduced $n$ variable Boolean SOP form will be termed a product term and each product term will be comprised of $n$ or fewer function variables. The structure of the product terms of a reduced Boolean SOP form are identical to the product terms, $\rho_{i}$, of the ESOP forms defined in chapter 2.

As the minterms of a Boolean function are mutually exclusive (or disjoint), it is possible to replace the inclusive-OR operator with the exclusive-OR operator without altering the operation of the expansion [3]. This forms an exclusive-OR (or ring) sum-of-products expansion of the function. The product terms of a reduced Boolean SOP form may or may not be disjoint. If each and every product term of the representation is disjoint, i.e. no overlapping product terms ( $\rho_{j} \rho_{k}=0$ for all $i \neq k, i, k=0,1, \ldots, 2^{n}-1$ ), then it is possible to form an ESOP expression directly from the reduced Boolean SOP form. That is, the inclusive-OR operator may be replaced by the exclusive-OR operator without altering the validity of the expression.

The following example illustrates the derivation of ESOP forms from reduced Boolean SOP forms. In the first equation the product terms of the Boolean function are disjoint whilst the Boolean SOP representation of the second function contains overlapping product terms.

Example 6.1 Convert the following Boolean functions to the equivalent ESOP forms.
1)

$$
\begin{align*}
f\left(x_{3}, x_{2}, x_{1}\right) & =\bar{x}_{3} x_{2} \bar{x}_{1}+x_{2} x_{1}+\bar{x}_{2} \\
& =\bar{x}_{3} x_{2} \bar{x}_{1} \oplus x_{2} x_{1} \oplus \bar{x}_{2} \tag{5.2}
\end{align*}
$$

The product terms of this reduced Boolean SOP form (Equation (5.2)) are disjoint (non-overlapping) hence it is valid to directly replace the inclusive-OR operator by the EXOR operator. The validity of this statement is illustrated by Equation (5.3). Each product term of the reduced Boolean SOP form is expanded to minterms and the OR operator is replaced by the EXOR operator. The rules of $G F(2)$ algebra are then employed to minimise the ESOP form. The product terms of the final ESOP form are equivalent to the initial representation.

$$
\begin{align*}
f\left(x_{3}, x_{2}, x_{1}\right) & =\bar{x}_{3} x_{2} \bar{x}_{1}+x_{2} x_{1}+\bar{x}_{2} \\
& =\bar{x}_{3} x_{2} \bar{x}_{1}+\bar{x}_{3} x_{2} x_{1}+x_{3} x_{2} x_{1}+\bar{x}_{3} \bar{x}_{2} \bar{x}_{1}+\bar{x}_{3} \bar{x}_{2} x_{1}+x_{3} \bar{x}_{2} \bar{x}_{1}+x_{3} \bar{x}_{2} x_{1} \\
& =\bar{x}_{3} x_{2} \bar{x}_{1} \oplus \bar{x}_{3} x_{2} x_{1} \oplus x_{3} x_{2} x_{1} \oplus \bar{x}_{3} \bar{x}_{2} \bar{x}_{1} \oplus \bar{x}_{3} \bar{x}_{2} x_{1} \oplus x_{3} \bar{x}_{2} \bar{x}_{1} \oplus x_{3} \bar{x}_{2} x_{1} \\
& =\bar{x}_{3} x_{2} \bar{x}_{1} \oplus x_{2} x_{1} \oplus \bar{x}_{2} \tag{5.3}
\end{align*}
$$

2) 

$$
\begin{equation*}
f\left(x_{3}, x_{2}, x_{1}\right)=\bar{x}_{3} x_{2}+x_{2} x_{1}+\bar{x}_{2} \tag{5.4}
\end{equation*}
$$

The product terms of this reduced Boolean SOP form (Equation (5.4)) are non-disjoint (overlapping), hence it is not valid to directly replace the inclusive-OR operator by the EXOR operator. Each product term of the reduced Boolean SOP form is expanded to the equivalent minterms (Equation (5.5)).

$$
\begin{align*}
f\left(x_{3}, x_{2}, x_{1}\right) & =\bar{x}_{3} x_{2}+x_{2} x_{1}+\bar{x}_{2} \\
& =\bar{x}_{3} x_{2} \bar{x}_{1}+\bar{x}_{3} x_{2} x_{1}+\bar{x}_{3} x_{2} x_{1}+x_{3} x_{2} x_{1}+\bar{x}_{3} \bar{x}_{2} \bar{x}_{1}+\bar{x}_{3} \bar{x}_{2} x_{1}+x_{3} \bar{x}_{2} \bar{x}_{1}+x_{3} \bar{x}_{2} x_{1} \tag{5.5}
\end{align*}
$$

Product term $\bar{x}_{3} x_{2} x_{1}$ occurs twice in the expanded form of Equation (5.5). According to the rules of Boolean algebra $x_{j}+x_{j}=x_{j}$, hence the second term may be removed forming the following equation.

$$
\begin{align*}
f\left(x_{3}, x_{2}, x_{1}\right) & =\bar{x}_{3} x_{2} \bar{x}_{1}+\bar{x}_{3} x_{2} x_{1}+x_{3} x_{2} x_{1}+\bar{x}_{3} \bar{x}_{2} \bar{x}_{1}+\bar{x}_{3} \bar{x}_{2} x_{1}+x_{3} \bar{x}_{2} \bar{x}_{1}+x_{3} \bar{x}_{2} x_{1} \\
& =\bar{x}_{3} x_{2} \bar{x}_{1} \oplus \bar{x}_{3} x_{2} x_{1} \oplus x_{3} x_{2} x_{1} \oplus \overline{3}_{3} \bar{x}_{2} \bar{x}_{1} \oplus \bar{x}_{3} \bar{x}_{2} x_{1} \oplus x_{3} \bar{x}_{2} \bar{x}_{1} \oplus x_{3} \bar{x}_{2} x_{1} \\
& =\bar{x}_{3} x_{2} \bar{x}_{1} \oplus x_{2} x_{1} \oplus \bar{x}_{2} \tag{5.6}
\end{align*}
$$

The product terms of the final ESOP form differ from the product terms of the initial Boolean function. This illustrates that it is not possible to directly generate an ESOP form from a reduced Boolean SOP containing overlapping product terms.
(End of example)

The algorithms described in the following sections may be employed to derive RM and KRO expansions from disjoint Boolean SOP forms. The product terms of the representation must be disjoint. It may be suggested
that this specification somewhat restricts the usefulness of these algorithms as the output of many Boolean minimisation packages, e.g. Espresso-II [63], is not a disjoint reduced Boolean SOP form. It is, however, possible to convert a reduced Boolean SOP form to a disjoint representation using a method such as that proposed by Falkowski and Perkowski [84]. Although this introduces an additional step into the conversion procedure the process of expanding the reduced Boolean SOP form to its equivalent canonical expression is avoided.
6.2.2 Extended Tabular Technique to Derive Positive Polarity Reed-Muller Expansions from Disjoint Reduced Boolean Sum-of-Products Forms
The tabular technique [1] reviewed in chapter 3 generates FPRM expansions from canonical Boolean SOP forms (each and every product term is a minterm). The method now proposed may be employed to derive PPRM expressions from reduced Boolean SOP forms where the product terms are disjoint. Henceforth, all reduced Boolean SOP forms will be presumed to be disjoint expressions unless otherwise stated.

The tabular notation must first be extended to allow the representation of product terms. The modified tabular notation introduced in chapter 5 will now be further developed. A reduced Boolean SOP form may be represented by a table where each row and column of that table represent a product term and variable of the expression, respectively. The columns will be headed using the notation $\hat{x}_{j}$ to represent a variable $x_{j}$ which may appear in both true and complemented forms throughout the expression. The variable is also positively biased. Consider function variable $x_{j}$ and product term $\rho_{i}$ of a reduced Boolean SOP form. Variable $x_{j}$ may be absent from term $\rho_{i}$ or present in either true or complemented form (but not both forms). Each cell of the table should contain either a 0 , a 1 or a -, indicating the state of each variable in the product term. Let the $n$-tuple $<c_{n} c_{n-1} \ldots c_{1}>$ denote a row of the table representing a reduced Boolean SOP form. Hence, any $c_{j}$ represents a cell of the table where $c_{j} \in\{0,1,-\}$ and $j=1,2, \ldots, n$. (It should be noted that $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle$ also denotes the condition of each variable in a product term of a reduced Boolean SOP form.) If literal $x_{j}\left(\bar{x}_{j}\right)$ appears in product term $\rho_{i}$ then column $j$ of the row representing $\rho_{i}$ should contain a $1(0)$ and the row is represented by the $n$-tuple $\left\langle c_{n} \ldots c_{j+1} l c_{j-1} \ldots c_{1}\right\rangle\left(\left\langle c_{n} \ldots c_{j+1} 0 c_{j-1} \ldots c_{1}\right\rangle\right)$. If, however, variable $x_{j}$ is
absent from product term $\rho_{i}$ then column $j$ of the row representing $\rho_{i}$ should contain a - . This row is represented by the $n$-tuple $<c_{n} \ldots c_{j+1}-c_{j-1}$ $\ldots c_{1}>$. $\left(0 \leq i \leq 2^{n}-1\right)$

The following example illustrates the use of the tabular notation to represent a reduced Boolean SOP form.

Example 6.2 Display the following 3 variable reduced Boolean SOP form using the tabular notation detailed previously.

Disjoint reduced Boolean SOP form

$$
\begin{equation*}
f\left(x_{3}, x_{2}, x_{1}\right)=\bar{x}_{3} x_{2} \bar{x}_{1} \oplus x_{2} x_{1} \oplus \bar{x}_{2} \tag{6}
\end{equation*}
$$

| $\hat{x}_{3}$ | $\hat{\boldsymbol{x}}_{2}$ | $\hat{x}_{1}$ |
| :--- | :--- | :--- |
| 0 | 1 | 0 |
| - | 1 | 1 |
| - | 0 | - |

(End of example)

The process of converting a Boolean function to a FPRM expansion may be considered to comprise of two distinct though not independent steps. Firstly, new product terms must be generated in which variables appear in the desired state (i.e. either true or complemented form) and in the second step any duplicate terms are cancelled. In order to generate a FPRM expansion from a reduced Boolean SOP form both these steps must be modified. These modifications are now introduced and a tabular technique for deriving the PPRM expansion is developed. In the following section this technique is extended to allow all KRO expansions to be generated from reduced Boolean SOP forms.

Consider the operation of generating the PPRM expansion of a Boolean function. Let the $n$-tuple $\left\langle e_{n} e_{n-1} \ldots e_{1}\right\rangle$ denote a product term generated from a row of the table representing a reduced Boolean $S O P$ form where $e_{j} \in$ $\{0,1\}, j=1,2, \ldots, n$.
If a minterm of the Boolean function is represented by a row of the form $<c_{n} \ldots c_{j+1} 0 c_{j-1} \ldots c_{1}>$ then a new row is generated which is represented by the
$n$-tuple $\left\langle e_{n} \ldots e_{j+1} 1 e_{j-1} \ldots e_{1}\right\rangle$ where $e_{l}=c_{l}$ for $l=1,2, \ldots, j-1, j+1, \ldots, n$. This operation may be expressed algebraically,

$$
\dot{x}_{n} \dot{x}_{n-1} \cdots \bar{x}_{f} \ldots \dot{x}_{1}=\dot{x}_{n} \dot{x}_{n-1} \ldots x_{f} \ldots \dot{x}_{1} \oplus \dot{x}_{n} \dot{x}_{n-1} \cdots \dot{x}_{j+1} \dot{x}_{j-1} \cdots \dot{x}_{1}
$$

Hence, the row $\left\langle c_{n} \ldots c_{j+1} 0 c_{j-1} \ldots c_{1}\right\rangle$ now represents the product term $\dot{x}_{n} \dot{x}_{n-1}$ $\ldots \dot{x}_{j+1} \dot{x}_{j-1} \ldots \dot{x}_{1}$ whilst the new product term $\dot{x}_{n} \dot{x}_{n-1} \ldots x_{j} \ldots \dot{x}_{1}$ is represented by the $n$-tuple $\left\langle e_{n} \ldots e_{j+1} 1 e_{j-1} \ldots e_{1}\right\rangle$. The new product term is compared with each product term (row) of the table. If the new product term (row) is equivalent to any existing term (row) then both the new term (row) and the existing term (row of the table) are deleted. Otherwise the new product term is added (modulo-2) to the expression i.e. a new row $\left\langle e_{n} \ldots e_{j+1} 1 e_{j-1} \ldots e_{1}\right\rangle$ is added to the table.

Now consider a reduced Boolean SOP form represented in tabular form. A product term is represented by the $n$-tuple $\left\langle c_{n} \ldots c_{j+1}-c_{j-1} \ldots c_{1}\right\rangle$. Consider the algebraic representation of this product term.

$$
\dot{x}_{n} \dot{x}_{n-1} \ldots \dot{x}_{j+1} \dot{x}_{j-1} \ldots \dot{x}_{1}=\dot{x}_{n} \dot{x}_{n-1} \ldots x_{J} \ldots \dot{x}_{1} \oplus \dot{x}_{n} \dot{x}_{n-1} \ldots \bar{x}_{f} \ldots \dot{x}_{1}
$$

The variable $x_{j}$ should be present only in true form throughout the PPRM expansion. Thus

$$
\begin{aligned}
\dot{x}_{n} \dot{x}_{n-1} \ldots \dot{x}_{j+1} \dot{x}_{j-1} \ldots \dot{x}_{1}= & \dot{x}_{n} \dot{x}_{n-1} \ldots x_{j} \ldots \dot{x}_{1} \\
& \oplus\left(\dot{x}_{n} \dot{x}_{n-1} \ldots x_{j} \ldots \dot{x}_{1} \oplus \dot{x}_{n} \dot{x}_{n-1} \ldots \dot{x}_{j+1} \dot{x}_{j-1} \ldots \dot{x}_{1}\right) \\
= & \dot{x}_{n} \dot{x}_{n-1} \ldots \dot{x}_{j+1} \dot{x}_{j-1} \ldots \dot{x}_{1}
\end{aligned}
$$

This indicates that if variable $x_{j}$ is absent from a product term then that product term is unaffected by the conversion procedure. This may be interpreted for the tabular technique. If a product term is represented by the $n$-tuple $\left\langle c_{n} \ldots c_{j+1}-c_{j-1} \ldots c_{1}\right\rangle$ then no new term is generated. However, the symbol - should be replaced by 0 and the row is now represented by the $n$-tuple $\left\langle c_{n} \ldots c_{j+1} 0 c_{j-1} \ldots c_{1}\right\rangle$. This is valid as the binary $n$-tuple represents a product term of the PPRM expansion, where a 0 indicates that a variable is absent from the product term and a 1 indicates that a variable is present in the product term.

The procedure for generating new terms (rows of the table) in the conversion from reduced Boolean SOP form to PPRM expansion has been detailed and the identification and removal of equivalent term is now reviewed. The product terms of a reduced Boolean SOP form are disjoint, however the process of generating new terms, as previously detailed, can form product terms which overlap with or, indeed, are contained by
product terms represented in the existing table. The process of identifying equivalent terms is no longer as simple a task as that performed in the original tabular technique.

Equivalent product terms may be identified by determining whether a newly generated product term and an existing product term intersect. If the product terms intersect then it is necessary to determine the product terms or minterms common to both the product terms tested. The common terms are removed and the product terms are modified. This procedure is now detailed where each product term is represented using the tabular notation.

The $n$-tuple $\left\langle e_{n} e_{n-1} \ldots e_{1}\right\rangle$ denotes a product term generated from a row of the table representing a reduced Boolean SOP form where $e_{j} \in\{0,1\}$ and $j=1,2, \ldots, n$. Consider a product term $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle$ of the existing table and product term $\left\langle e_{n} e_{n-1} \ldots e_{1}\right\rangle$ which was generated from a different product term of that table. It is necessary to determine whether $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle$ and $<e_{n} e_{n-1} \ldots e_{1}>$ intersect (contain common product terms). Intersection may be determined from the following table.

|  |  | $c_{j}$ |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  | $n$ | 0 | 1 | - |  |
|  | 0 | 0 | $\emptyset$ | 0 |  |
| $e_{j}$ | 1 | $\emptyset$ | 1 | 1 |  |
|  | - | 0 | 1 | - |  |

Let $\emptyset$ denote the empty set and $\cap$ denote intersection.
If any $c_{j} \cap e_{j}=\emptyset(j \in\{1,2, \ldots, n\})$ then the product terms represented by $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle$ and $\left\langle e_{n} e_{n-1} \ldots e_{1}\right\rangle$ do not intersect. Therefore, no product terms can be deleted.
Iff all $c_{j} \cap e_{j} \neq \emptyset(j=1,2, \ldots, n)$ then the product terms represented by $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle$ and $\left\langle e_{n} e_{n-1} \ldots e_{1}\right\rangle$ intersect. It is necessary to determine the common product terms and remove these terms from $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle$ and $\left\langle e_{n} e_{n-1}\right.$ $\ldots e_{1}>$. Thus, the modified form of the existing product term $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle$ is equal to the product term(s) formed from

$$
\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle \cap\left(\overline{\left.\left\langle c_{n} c_{n-1} \cdots c_{1}\right\rangle \cap<e_{n} e_{n-1} \cdots e_{1}\right\rangle}\right)
$$

Similarly, the modified form of the generated product term $\left\langle e_{n} e_{n-1} \ldots e_{1}\right\rangle$ is equal to the product term(s) formed from

$$
\left.<e_{n} e_{n-1} \ldots e_{1}\right\rangle \cap\left(\overline{\left.\left\langle c_{n} c_{n-1} \cdots c_{1}\right\rangle \cap<e_{n} e_{n-1} \cdots e_{1}\right\rangle}\right)
$$

Note, when determining the intersections the rules of Boolean algebra apply.
A method for deriving these modified forms is now proposed.

## Cancelling equivalent product terms

S. 1 For $j=1,2, \ldots, n$, count the number of occasion where $c_{j}=-$ and $e_{j}$ $=0$ or 1 . Let this equal diff. This quantity is the number of product terms required to represent $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle$ after the common product terms have been removed.
S. 2 Construct diff product terms denoted $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle_{1},\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle_{2}, \ldots$, $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle_{d i f f}$ Let $k=\ell=1$.
S. 3 For $j=1,2, \ldots, n$,
S.3a Evaluating $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle$ and $\left\langle e_{n} e_{n-1} \ldots e_{1}\right\rangle$, if $c_{j}=e_{j}$, or $c_{j}=0(1)$ and $e_{j}=-$, then let each $c_{j}$ of $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle_{\ell}$ equal $c_{j}$ of $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle \quad(\ell=$ $1,2, \ldots$, diff $)$. Otherwise go to $\mathrm{S.3b}$
S.3b If $c_{j}$ of $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle$ equals - and $e_{j}$ of $\left\langle e_{n} e_{n-1} \ldots e_{1}\right\rangle$ equals 0 (1) then $c_{j}$ of $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle_{1},\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle_{2}, \ldots,\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle_{k-1}$, equals and $c_{j}$ of $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle_{k}$ equals 1(0). Each $c_{j}$ of $\left\langle c_{n} c_{n-1} \cdots c_{1}\right\rangle_{k+1}$, $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle_{k+2}, \ldots,\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle_{d i f f}$ equals $e_{j}$ of $\left\langle e_{n} e_{n-1} \ldots e_{1}\right\rangle$. Let $k=k+1$.
S. 4 Generate the product terms formed from $\left\langle e_{n} e_{n-1} \ldots e_{1}\right\rangle$ after the removal of common product terms. Repeat S.1-S.3, replacing every $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle$ by $\left\langle e_{n} e_{n-1} \ldots e_{1}\right\rangle$ and vice versa. Each $c_{j}$ and $e_{j}$ should be interchanged and the new product terms are denoted $\left\langle e_{n} e_{n-1} \ldots e_{1}\right\rangle_{1}$, $\left.<e_{n} e_{n-1} \ldots e_{1}\right\rangle_{2}, \ldots,\left\langle e_{n} e_{n-1} \ldots e_{1}\right\rangle_{d i f f}$ diff indicates the number of occasion where $e_{j}=-$ and $c_{j}=0$ or 1.

This method derives the product terms formed from $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle$ and $\left\langle e_{n} e_{n-1}\right.$ $\left.\ldots e_{1}\right\rangle$ when product terms common to both $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle$ and $\left\langle e_{n} e_{n-1} \ldots e_{1}\right\rangle$ are deleted.

The generation and cancellation operations of the tabular technique have been modified to include product terms as well as the minterms used in the
original form. The following algorithm may be employed to generated PPRM expansions from reduced Boolean SOP forms.

## Converting a reduced Boolean SOP form to a PPRM expansion

S. 1 Represent the reduced $n$ variable Boolean function using the tabular notation. Thus, for $j=1, \ldots, n$ form a column with the heading $\hat{x}_{j}$, i.e. each function variable is present in both true and complemented forms and each variable is positively biased. Form the rows of the table where each row represents a product term. Each cell of the table is filled with 0,1 or - according to the rules defined previously. A $n$-tuple $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle$ denotes each row of the table.
Let $h=n$.
S. 2 Let $j=h$.

If a row of the table (product term of the Boolean SOP form) has a representation of the form $\left\langle c_{n} \ldots c_{j+1} 0 c_{j-1} \ldots c_{1}\right\rangle$ then generate a new row which is represented by the $n$-tuple $\left\langle e_{n} \ldots e_{j+1} 1 e_{j-1} \ldots e_{1}>\right.$ where $e_{\ell}$ $=c_{l}$ for $\ell=1,2, \ldots, j-1, j+1, \ldots, n$. Go to S.3.
Otherwise, if a row of the table (product term of the Boolean SOP form) has a representation of the form $\left\langle c_{n} \ldots c_{j+1}-c_{j-1} \ldots c_{1}\right\rangle$ then change this representation to $\left\langle c_{n} \ldots c_{j+1} 0 c_{j-1} \ldots c_{1}\right\rangle$ and go to S.5.
S. 3 Test for intersection between each row of the table and the newly generated term:-
If any $c_{j} \cap e_{j}=\emptyset(j \in\{1,2, \ldots, n\})$ then the product terms represented by $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle$ and $\left\langle e_{n} e_{n-1} \ldots e_{1}\right\rangle$ do not intersect. Add $\left\langle e_{n} e_{n-1} \ldots e_{1}\right\rangle$ to the foot of the existing table. Go to S.5.
Iff all $c_{j} \cap e_{j} \neq \emptyset(j=1,2, \ldots, n)$ then the product terms represented by $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle$ and $\left\langle e_{n} e_{n-1} \ldots e_{1}\right\rangle$ intersect. Go to S.4.
S. 4 Employ the algorithm detailed previously, namely, cancelling equivalent product terms. This removes equivalent terms from the existing and newly generated terms. Hence, $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle$ is deleted and the modified rows $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle_{1},\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle_{2}, \ldots,\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle_{\text {diff }}$ are added to the foot of the existing table (remove the subscripts $1,2, \ldots$, diff $)$. Each of the product terms $\left\langle e_{n} e_{n-1} \ldots e_{1}\right\rangle_{1},\left\langle e_{n} e_{n-1} \ldots e_{1}\right\rangle_{2}, \ldots$, $\left.<e_{n} e_{n-1} \ldots e_{1}\right\rangle_{d i f f}$ formed by modifying the generated product terms is compared again with the existing table to ensure all equivalent terms are deleted. Therefore, go to S.3, the algorithm to cancel equivalent terms is repeatedly applied to each $\left\langle e_{n} e_{n-1} \ldots e_{1}\right\rangle_{1},\left\langle e_{n} e_{n-1} \ldots e_{1}\right\rangle_{2}, \ldots$,
$\left.<e_{n} e_{n-1} \ldots e_{1}\right\rangle_{d i f f}$ until no intersections are determined. Add all of the remaining terms to the foot of the table.
S. 5 If the current row is the last row in the table then go to S.6, otherwise go to S.2.
S. 6 New column heading is $x_{f}$
S. 7 If $h>1$ then let $h=h-1$ and go to S.2. Otherwise the table represents the PPRM expansion.

The PPRM expansion may be constructed from the final table generated by the preceding algorithm. The rules for interpreting the tabular representation have been developed in chapter 5. It is now possible to convert the PPRM expansion to any FPRM expansion using the tabular technique described in [1] and reviewed in chapter 3.

The following example illustrates the derivation of the PPRM expansion of a reduced Boolean SOP form using the above algorithm.

Example 6.3 Derive the RM expansion of a 4 variable Boolean function expressed in disjoint SOP form.

$$
f\left(x_{4}, x_{3}, x_{2}, x_{1}\right)=\bar{x}_{4} \bar{x}_{3} x_{2} x_{1}+\bar{x}_{4} x_{3} \bar{x}_{1}+x_{3} \bar{x}_{2} x_{1}+x_{4} \bar{x}_{3} \bar{x}_{2}+x_{4} x_{3} x_{2}
$$

Transform the Boolean SOP form to the equivalent RM expansion (polarity 0 FPRM expansion), applying the transformation rules.
( ${ }_{i}$ denotes equivalent or intersecting terms)
Generated terms

Boolean function

|  | $\hat{x}_{4}$ | $\hat{x}_{3}$ | $\hat{x}_{2}$ | $\hat{x}_{1}$ |
| :--- | :--- | :--- | :--- | :--- |
| 0 | 0 | 1 | 1 |  |
| 0 | 1 | - | 0 |  |
|  | - | 1 | 0 | 1 |
|  | 1 | 0 | 0 | - |
| ${ }^{1}$ | 1 | 1 | 1 | - |


|  | $\left(x_{4}\right)$ |  |  |  |  |
| :--- | :--- | :--- | :--- | :--- | :---: |
|  | $x_{4}$ | $\hat{x}_{3}$ | $\hat{x}_{2}$ | $\hat{x}_{1}$ |  |
|  | 1 | 0 | 1 | 1 |  |
| ${ }^{1}$ | 1 | 1 | - | 0 |  |

Generated terms

\[

\]

Generated terms

|  | $\left(x_{2}\right)$ |  |  |  |  |
| :--- | :--- | :--- | :--- | :--- | :---: |
|  | $x_{4}$ | $x_{3}$ | $x_{2}$ | $\hat{x}_{1}$ |  |
|  | 0 | 1 | 1 | 1 |  |
| ${ }^{*}$ | 1 | 0 | 1 | - |  |
|  | 1 | 1 | 1 | 1 |  |

Generated terms

$$
\left(x_{1}\right)
$$

$$
\begin{array}{llll}
x_{4} & x_{3} & x_{2} & \hat{x}_{1} \\
\hline 0 & 0 & 1 & 1
\end{array}
$$

$$
\begin{array}{llll}
0 & 1 & 0 & 0
\end{array}
$$

$$
{ }_{6} \begin{array}{lllll} 
& 0 & 1 & 0 & 1
\end{array}
$$

$$
\begin{array}{llll}
1 & 0 & 0 & -
\end{array}
$$

$$
\begin{array}{llll}
1 & 1 & 0 & 1
\end{array}
$$

$$
\begin{array}{llll}
1 & 0 & 1 & 0
\end{array}
$$

$$
\begin{array}{llll}
1 & 1 & 1 & 1
\end{array}
$$

RM expansion

| $x_{4}$ | $x_{3}$ | $x_{2}$ | $x_{1}$ |  |
| :--- | :--- | :--- | :--- | :--- |
| 0 | 0 | 1 | 1 | $\left(x_{2} x_{1}\right)$ |
| 0 | 1 | 0 | 0 | $\left(x_{3}\right)$ |
| 1 | 0 | 0 | 0 | $\left(x_{4}\right)$ |
| 1 | 1 | 0 | 1 | $\left(x_{4} x_{3} x_{1}\right)$ |
| 1 | 0 | 1 | 0 | $\left(x_{4} x_{2}\right)$ |
| 1 | 1 | 1 | 1 | $\left(x_{4} x_{3} x_{2} x_{1}\right)$ |
| 1 | 0 | 1 | 1 | $\left(x_{4} x_{2} x_{1}\right)$ |

The Boolean function has been transformed to the equivalent RM expansion and is represented by the following equation.

$$
f_{0}\left(x_{4}, x_{3}, x_{2}, x_{1}\right)=x_{2} x_{1} \oplus x_{3} \oplus x_{4} \oplus x_{4} x_{2} \oplus x_{4} x_{2} x_{1} \oplus x_{4} x_{3} x_{1} \oplus x_{4} x_{3} x_{2} x_{1}
$$

(End of example)

The above algorithm was implemented in Pascal and the programs executed on a Dell P60 personal computer.

### 6.2.3 Extended Tabular Technique to Derive Kronecker Expansions from Disjoint Reduced Boolean Sum-of-Products Forms

The tabular technique described in the preceding section may be employed to generate the PPRM expansion of a reduced Boolean SOP form. Any FPRM expansion may then be derived from this PPRM expression using the technique described in [1]. It is, however, more efficient to derive a FPRM expansion, or indeed a KRO expression, directly from the initial Boolean representation. The tabular techniques presented in chapter 5 may be used to generate KRO expansions from Boolean functions. The method presented in this section may be used to generate any KRO expansion from a reduced Boolean SOP form. It is an amalgamation of the notation and algorithms introduced in chapter 5 and the algorithm presented in section 6.2.2 of this chapter. Some preliminary discussion is necessary before the new algorithm is presented.

Consider a reduced Boolean SOP form represented in tabular form. Any product term is represented by the $n$-tuple $\left\langle c_{n} \ldots c_{j+1}-c_{j-1} \ldots c_{1}\right\rangle$. Now, consider the algebraic representation of this product term.

$$
\dot{x}_{n} \dot{x}_{n-1} \ldots \dot{x}_{j+1} \dot{x}_{j-1} \ldots \dot{x}_{1}=\dot{x}_{n} \dot{x}_{n-1} \ldots x_{f} \ldots \dot{x}_{1} \oplus \dot{x}_{n} \dot{x}_{n-1} \ldots \bar{x}_{f} \ldots \dot{x}_{1}
$$

If a variable $x_{j}$ is present only in true form throughout a KRO expansion then the variable takes the form that it would adopt in the PPRM expansion. This particular case has been discussed in the previous section and is valid for the generation of KRO expressions. Hence, the product term would be represented by the $n$-tuple $\left\langle c_{n} \ldots c_{j+1} 0 c_{j-1} \ldots c_{1}\right\rangle$ in the table representing the KRO expansion.
If a variable $x_{j}$ is present only in complemented form throughout a KRO expansion then

$$
\begin{gathered}
\dot{x}_{n} \dot{x}_{n-1} \ldots \dot{x}_{j+1} \dot{x}_{j-1} \ldots \dot{x}_{1}=\left(\dot{x}_{n} \dot{x}_{n-1} \ldots \bar{x}_{f} \ldots \dot{x}_{1} \oplus \dot{x}_{n} \dot{x}_{n-1} \ldots \dot{x}_{j+1} \dot{x}_{j-1} \ldots \dot{x}_{1}\right) \\
\oplus \dot{x}_{n} \dot{x}_{n-1} \ldots \bar{x}_{f} \ldots \dot{x}_{1} \\
=\dot{x}_{n} \dot{x}_{n-1} \ldots \dot{x}_{j+1} \dot{x}_{j-1} \cdots \dot{x}_{1}
\end{gathered}
$$

This indicates that if variable $x_{j}$ is absent from a product term then that product term is unaffected by the conversion procedure. This may be interpreted for the tabular technique. If a product term is represented by the $n$-tuple $\left\langle c_{n} \ldots c_{j+1}-c_{j-1} \ldots c_{1}>\right.$ then no new term is generated. However, the symbol - should be replaced by 1 (the variable is now negatively biased) and the row is now represented by the $n$-tuple $\left\langle c_{n} \ldots c_{j+1} 1 c_{j-1} \ldots c_{1}\right\rangle$. If a variable $x_{j}$ is present in both true and complemented forms throughout a KRO expansion then

$$
\dot{x}_{n} \dot{x}_{n-1} \ldots \dot{x}_{j+1} \dot{x}_{j-1} \ldots \dot{x}_{1}=\dot{x}_{n} \dot{x}_{n-1} \ldots x_{f} \ldots \dot{x}_{1} \oplus \dot{x}_{n} \dot{x}_{n-1} \ldots \bar{x}_{f} \ldots \dot{x}_{1}
$$

This indicates that if variable $x_{j}$ is absent from a product term then that product term must be replaced by two new product terms. One of these product terms should contain the literal $x_{j}$ whilst the other should contain the literal $\bar{x}_{j}$. This may be interpreted for the tabular technique. If a product term is represented by the $n$-tuple $\left\langle c_{n} \ldots c_{j+1}-c_{j-1} \ldots c_{1}\right\rangle$ then replace $\left\langle c_{n} \ldots c_{j+1}-c_{j-1} \ldots c_{1}\right\rangle$ with $\left\langle c_{n} \ldots c_{j+1} 0 c_{j-1} \ldots c_{1}\right\rangle$. Generate a new row of the table with the representation $\left\langle e_{n} \ldots e_{j+1} l e_{j-1} \ldots e_{1}\right\rangle$ where $e_{l}=c_{l}, l=1,2, \ldots, j 1$, $j+1, \ldots, n$.

Converting a reduced Boolean SOP form to a polarity m KRO expansion
S. 1 Represent the $n$ variable Boolean function using the tabular notation. Thus for $j=1, \ldots, n$ form a column with the heading $\hat{x}_{j}$, i.e. each function variable is present in both true and complemented forms
and each variable is positively biased. Form the rows of the table where each row represents a minterm. Each cell of the table is filled with 0,1 or - according to the rules defined previously. A n-tuple $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle$ denotes each row of the table. Let $h=n$.
S. 2 Let $j=h$.

If $m_{j}=0\left(m_{j}=1\right)$ then
if a row of the table has a representation of the form $<c_{n} \ldots c_{j+1} 0 c_{j-1} \ldots c_{1}>\left(\left\langle c_{n} \ldots c_{j+1} 1 c_{j-1} \ldots c_{1}>\right)\right.$ then generate a new row which is represented by the $n$-tuple $\left\langle e_{n} \ldots e_{j+1} 1 e_{j-1} \ldots e_{1}\right\rangle$ ( $\left\langle e_{n} \ldots e_{j+1} 0 e_{j-1} \ldots e_{1}\right\rangle$ ) where $e_{\ell}=c_{\ell}$ for $\ell=1,2, \ldots, j-1, j+1, \ldots, n$. Go to S.3.
Otherwise, if a row of the table has a representation of the form $\left\langle c_{n} \ldots c_{j+1}-c_{j-1} \ldots c_{1}\right\rangle$ then alter this representation to $\left.<c_{n} \ldots c_{j+1} 0 c_{j-1} \ldots c_{1}\right\rangle\left(\left\langle c_{n} \ldots c_{j+1} 1 c_{j-1} \ldots c_{1}>\right)\right.$ and go to S.5.
If $m_{j}=2$ then
if a row of the table has a representation of the form $\left\langle c_{n} \ldots c_{j+1} 0 c_{j-1} \ldots c_{1}\right\rangle\left(\left\langle c_{n} \ldots c_{j+1} 1 c_{j-1} \ldots c_{1}\right\rangle\right)$ then do not generate any new terms and go to S.5.
Otherwise, if a row of the table has a representation of the form $\left\langle c_{n} \ldots c_{j+1}-c_{j-1} \ldots c_{1}\right\rangle$ then alter this representation to $<c_{n} \ldots c_{j+1} 0 c_{j-1} \ldots c_{1}>$. Generate a new product term with the representation $\left\langle e_{n} \ldots e_{j+1} 1 e_{j-1} \ldots e_{1}\right\rangle$ where $e_{\ell}=c_{\ell}$ for $\ell=1,2, \ldots$, $j-1, j+1, \ldots, n$. Add this new term to the foot of the table and go to S.5.
S. 3 Test for intersection between each row of the table and the newly generated term:-
If any $c_{j} \cap e_{j}=\emptyset(j \in\{1,2, \ldots, n\})$ then the product terms represented by $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle$ and $\left\langle e_{n} e_{n-1} \ldots e_{1}\right\rangle$ do not intersect. Add $\left\langle e_{n} e_{n-1} \ldots e_{1}\right\rangle$ to the foot of the existing table. Go to S.5.
Iff all $c_{j} \cap e_{j} \neq \emptyset(j=1,2, \ldots, n)$ then the product terms represented by $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle$ and $\left\langle e_{n} e_{n-1} \ldots e_{1}\right\rangle$ intersect. Go to S.4.
S. 4 Employ the algorithm detailed previously, namely, cancelling equivalent product terms. This removes equivalent terms from the existing and newly generated terms. Hence, $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle$ is deleted and the modified rows $\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle_{1},\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle_{2}, \ldots,\left\langle c_{n} c_{n-1} \ldots c_{1}\right\rangle$ diff are added to the foot of the existing table (remove the subscripts $1,2, \ldots, d i f f)$. Each of the product terms $\left\langle e_{n} e_{n-1} \ldots e_{1}\right\rangle_{1},\left\langle e_{n} e_{n-1} \ldots e_{1}\right\rangle, \ldots$,
$\left.<e_{n} e_{n-1} \ldots e_{1}\right\rangle_{\text {diff }}$ formed by modifying the generated product terms is compared again with the existing table. This is to ensure all equivalent terms are deleted. Therefore, go to S.3, the algorithm to cancel equivalent terms is repeatedly applied to each $\left\langle e_{n} e_{n-1} \ldots e_{1}\right\rangle_{1}$, $\left.\left.<e_{n} e_{n-1} \ldots e_{1}\right\rangle_{2}, \ldots,<e_{n} e_{n-1} \ldots e_{1}\right\rangle_{d i r f}$ until no intersections are determined. Add all of the remaining terms to the foot of the table.
S. 5 If the current row is the last row in the table then go to S.6, otherwise go to S.2.
S. 6 If $m_{j}=0$ then
alter column heading according to the polarity of the variable.
New column heading is $x_{f}$
If $m_{j}=1$ then
alter column heading according to the polarity of the variable and change the bias of the variable. New column heading is $\underline{\underline{X}}_{j}$. If $m_{j}=2$ then column heading does not change.
S. 7 If $h>1$ then let $h=h-1$ and go to S.2. Otherwise the table represents the polarity $m$ KRO expansion. The state of each variable is indicated by the header row of the table.

The polarity $m$ KRO expansion may be constructed from the final table generated by the preceding algorithm. The rules for interpreting the tabular representation have been developed in chapter 5.

The following example illustrates the generation of a KRO expansion from a disjoint Boolean SOP form. The algorithm described above is employed to perform the conversion.

Example 6.4 Derive the polarity 69 KRO expansion of a 4 variable Boolean function expressed in disjoint SOP form.

$$
f\left(x_{4}, x_{3}, x_{2}, x_{1}\right)=\bar{x}_{4} \bar{x}_{3} x_{2} x_{1}+\bar{x}_{4} x_{3} \bar{x}_{1}+x_{3} \bar{x}_{2} x_{1}+x_{4} \bar{x}_{3} \bar{x}_{2}+x_{4} x_{3} x_{2}
$$

Transform the Boolean SOP form to the polarity 69 KRO expansion, applying the transformation rules.
( ${ }_{1}$ denotes equivalent or intersecting terms)

Generated terms

Boolean function

| $\hat{x}_{4}$ | $\hat{X}_{3}$ | $\hat{x}_{2}$ | $\hat{x}_{1}$ |
| :--- | :--- | :--- | :--- |
| 0 | 0 | 1 | 1 |

$01-0$

- 1001
$100-$
111 -

| $\left(\hat{x}_{4}\right)$ |  |  |  |
| :--- | :--- | :--- | :--- |
| $\hat{x}_{4}$ | $\hat{x}_{3}$ | $\hat{x}_{2}$ | $\hat{x}_{1}$ |
| 1 | 1 | 0 | 1 |

Generated terms
$\left(\overline{\underline{X}}_{3}\right)$

| $\hat{x}_{4}$ | $\overline{\underline{X}}_{3}$ | $\hat{x}_{2}$ | $\hat{x}_{1}$ |
| :--- | :--- | :--- | :--- |
| 0 | 0 | - | 0 |

$\begin{array}{llll}0 & 0 & 0 & 1\end{array}$
101
$\begin{array}{lllll}* & 1 & 0 & 0 & 1\end{array}$

Generated terms
$\left(\hat{x}_{2}\right)$

| $\hat{x}_{4}$ | $\underline{\underline{x}}_{3}$ | $\hat{X}_{2}$ | $\hat{x}_{1}$ |
| :--- | :--- | :--- | :--- |
| 0 | 1 | 1 | 0 |

$\begin{array}{llll}0 & 0 & 1 & 0\end{array}$
$\begin{array}{llll}0 & 1 & 0 & 1\end{array}$
1000
111 -
$\begin{array}{llll}1 & 1 & 0 & 1\end{array}$
$00-0$
$\begin{array}{llll}0 & 0 & 0 & 1\end{array}$
101 -

## Generated terms



Polarity 69 KRO expansion

| $\hat{x}_{4}$ | $\underline{\underline{x}}_{3}$ | $\hat{x}_{2}$ | $x_{1}$ |  |
| :--- | :--- | :--- | :--- | :--- |
| 0 | 1 | 0 | 0 | $\left(\bar{x}_{4} \bar{x}_{2}\right)$ |
| 1 | 0 | 0 | 0 | $\left(x_{4} \bar{x}_{3} \bar{x}_{2}\right)$ |
| 1 | 1 | 1 | 0 | $\left(x_{4} x_{2}\right)$ |
| 1 | 1 | 0 | 1 | $\left(x_{4} \bar{x}_{2} x_{1}\right)$ |
| 0 | 0 | 0 | 0 | $\left(\bar{x}_{4} \bar{x}_{3} \bar{x}_{2}\right)$ |
| 1 | 0 | 1 | 0 | $\left(x_{4} \bar{x}_{3} x_{2}\right)$ |
| 0 | 1 | 1 | 0 | $\left(\bar{x}_{4} x_{2}\right)$ |
| 0 | 0 | 1 | 0 | $\left(\bar{x}_{4} \bar{x}_{3} x_{2}\right)$ |
| 1 | 0 | 0 | 1 | $\left(x_{4} \bar{x}_{3} \bar{x}_{2} x_{1}\right)$ |
| 0 | 1 | 1 | 1 | $\left(\bar{x}_{4} x_{2} x_{1}\right)$ |

The Boolean function has been transformed to the equivalent polarity 69 KRO expansion and is represented by the following equation.

$$
\begin{aligned}
f_{\mathscr{A}}\left(x_{4} x_{3}, x_{2}, x_{1}\right)= & x_{4} x_{2} \oplus x_{4} \bar{x}_{2} x_{1} \oplus x_{4} \bar{x}_{3} x_{2} \oplus x_{4} \bar{x}_{3} \bar{x}_{2} \oplus x_{4} \bar{x}_{3} \bar{x}_{2} x_{1} \\
& \oplus \bar{x}_{4} x_{2} \oplus \bar{x}_{4} x_{2} x_{1} \oplus \bar{x}_{4} \bar{x}_{2} \oplus \bar{x}_{4} \bar{x}_{3} \bar{x}_{2} \bar{x}_{4} \bar{x}_{2}
\end{aligned}
$$

(End of example)
The above algorithm was implemented in Pascal and the programs executed on a Dell P60 personal computer.

### 6.3 Summary

The algorithms presented in this chapter provide a means of deriving RM and KRO expansions from disjoint reduced Boolean SOP forms. The algorithms are an efficient approach as the Boolean SOP forms may be converted directly to the RM or KRO expressions without the need to generate the minterms of the Boolean function, a costly operation both in the memory requirements and processor time. Additionally, the development of these algorithms has significantly increased the usefulness of the tabular technique making it a simple and valuable method for deriving KRO expansions from Boolean SOP forms.

## Chapter 7

## Graphical Representation of Fixed Polarity Reed-Muller Expansions'

Reed-Muller expansions provide an alternative means of representing switching function, as has been demonstrated in the preceding chapters of this thesis. Each switching function has been denoted in two-level form i.e. as SOP or ESOP expressions, the multiple level (multi-level) representation of switching functions is now described.

Two-level logic minimisation techniques for Boolean functions derive minimal implementations for PLAs, devices which directly implement SOP forms [63]. However, the constraints imposed by two-level representations may lead to inefficiencies. That is, the resulting circuit may be large, performance may not meet specifications or the two-level representation may not be realisable in the target technology. This has led to the development of techniques for representing and optimising multi-level combinational logic circuits. A multi-level circuit has two or more levels of logic between the primary inputs and primary outputs. Indeed, two-level logic functions are a special case of multi-level logic functions. Multi-level logic circuits are more traditionally known as random logic and offer freedom to restructure and optimise away from the constraints imposed by two-level circuits. This freedom does, however, mean that more sophisticated techniques are required in order to determine efficient implementations of switching functions.

The following section of this chapter reviews techniques for representing Boolean SOP forms as multi-level combinational logic circuits. This includes a discussion of the existing methods used to construct efficient Binary Decision Diagrams (section 2.6 of chapter 2). Additionally, the Reed-Muller factored form, developed by Saul [85], is described. Alternative graphical representations of FPRM expansions called Reed-Muller Binary Decision

[^2]Diagrams are described in section 7.2. RMBDDs are a counterpart of the conventional BDDs and may be used to directly implement FPRM expressions as multi-level circuits. These graphical structures are initially given as full trees, with restrictions placed on the ordering of variables, and are defined as Ordered Reed-Muller Binary Decision Trees. Rules for reducing ORMBDTs and hence forming Reduced Ordered Reed-Muller Binary Decision Diagrams are described. Similar to BDDs, the size of any RORMBDD is very sensitive to the choice of variable ordering. An algorithm is detailed which finds a good variable order with respect to the number of nodes in the RORMBDDs. Additionally, the physical implementation of RORMBDDs using Reed-Muller Universal Logic Modules (RM-ULMs) [88, 89] is discussed.

### 7.1 Multi-level Logic Synthesis Techniques

A multi-level representation of a switching function may be defined recursively as a sum of products of sums of products..... of arbitrary depth, or alternatively, as a product of sums of products of sums..... of arbitrary depth [33, 90, 91].

For example,

$$
\begin{aligned}
f\left(x_{5}, x_{4}, x_{3}, x_{2}, x_{1}\right) & =\left(\bar{x}_{2} x_{1}+x_{3} x_{2}\right)\left(x_{3}+\bar{x}_{4}\right)+x_{4}\left(x_{1}+x_{5} x_{2}\right) \\
g\left(x_{4}, x_{3}, x_{2}, x_{1}\right) & =\left(x_{1}+x_{2}\left(x_{3}+\bar{x}_{4}\right)\right)\left(\bar{x}_{2}+x_{4}\right)
\end{aligned}
$$

These expressions are known as the factored forms of a function. The factored form may be directly implemented to give a multi-level circuit performing the required function.

Many familiar methods suitable for expressing and minimising Boolean functions e.g. Karnaugh and Veitch maps and the Quine-McCluskey technique, are applicable only to two-level representations of the Boolean function. This has necessitated the development of techniques suitable for representing and optimising the multi-level forms of Boolean functions. A number of these techniques are detailed in the following literature review. Initially, algebraic techniques for deriving multi-level representations of Boolean functions are described.

The goals of synthesis and optimisation techniques suitable for multi-level
representations include efficient use of area (compact representation), reduction of delays from primary inputs to primary outputs and maximising the testability of the multi-level representation. The basic approach adopted by most multi-level logic synthesis systems is that of restructuring a logic function so as to determine an optimal (sub-optimal) multi-level structure, then employing minimisation techniques to optimally (sub-optimally) represent nodes of the overall structure. The restructuring or resynthesis of logic functions employs techniques known as extraction, collapsing, simplification, substitution, factoring and decomposition.

## Extraction

The identification of common subexpressions and generation of new intermediate variables. New variables replace the existing subexpressions. e.g.

$$
\begin{aligned}
f\left(x_{4}, x_{3}, x_{2}, x_{1}\right) & =\left(x_{1}+\bar{x}_{2}\right) x_{3}+x_{4} \\
g\left(x_{5}, x_{4}, x_{3}, x_{2}, x_{1}\right) & =\left(x_{1}+\bar{x}_{2}\right) x_{3}
\end{aligned}
$$

Extraction yields

$$
\begin{aligned}
f\left(x_{4}, x_{3}, x_{2}, x_{1}\right) & =y\left(x_{2}, x_{1}\right) x_{3}+x_{4} \\
g\left(x_{5}, x_{4}, x_{3}, x_{2}, x_{1}\right) & =y\left(x_{2}, x_{1}\right) x_{5} \\
y\left(x_{2}, x_{1}\right) & =x_{1}+\overline{x_{2}}
\end{aligned}
$$

## Collapsing

This is, effectively, the inverse of extraction as intermediate variables are expanded into the subexpressions they represent. This operation has the potential to reduce the number of nodes in the network.
e.g.

$$
\begin{aligned}
f\left(x_{4}, x_{3}, x_{2}, x_{1}\right) & =y\left(x_{2}, x_{1}\right) x_{3}+\bar{y}\left(x_{2}, x_{1}\right) x_{4} \\
y\left(x_{2}, x_{1}\right) & =x_{1}+\overline{x_{2}}
\end{aligned}
$$

Collapsing $y\left(x_{2}, x_{1}\right)$ back into $f\left(x_{4}, x_{3}, x_{2}, x_{1}\right)$ yields

$$
f\left(x_{4}, x_{3}, x_{2}, x_{1}\right)=x_{3} x_{1}+x_{3} \bar{x}_{2}+x_{4} x_{2} \bar{x}_{1}
$$

## Simplification

Replacing nodes by equivalent but simpler expressions. Implemented by two-level minimisers employing the 'don't care' terms of the multi-level circuit.

## Substitution

Expressing the function represented by one node in the network by the function represented by another node in the network.
e.g.

$$
\begin{aligned}
y\left(x_{2}, x_{1}\right) & =x_{1}+x_{2} \\
f\left(x_{3}, x_{2}, x_{1}\right) & =x_{1}+x_{3} x_{2}
\end{aligned}
$$

Substituting $y\left(x_{2}, x_{1}\right)$ into $f\left(x_{3}, x_{2}, x_{1}\right)$ yields

$$
f\left(x_{3}, x_{2}, x_{1}\right)=y\left(x_{2}, x_{1}\right)\left(x_{1}+x_{3}\right)
$$

## Factoring

Reducing the complexity of individual nodes in the network by determining optimal (sub-optimal) factored forms.
e.g.

$$
f\left(x_{5}, x_{4}, x_{3}, x_{2}, x_{1}\right)=x_{3} x_{1}+x_{3} x_{2}+x_{4} x_{1}+x_{4} x_{2}+x_{5}
$$

The factored form of $f\left(x_{5}, x_{4}, x_{3}, x_{2}, x_{1}\right)$ is

$$
f\left(x_{3}, x_{4}, x_{3}, x_{2}, x_{1}\right)=\left(x_{1}+x_{2}\right)\left(x_{3}+x_{4}\right)+x_{3}
$$

## Decomposition

Reducing the complexity of the network by factoring. Factors are realised as intermediate variables and form new nodes in the network. e.g.

$$
f\left(x_{4}, x_{3}, x_{2}, x_{1}\right)=x_{3} x_{2} x_{1}+x_{4} x_{2} x_{1}
$$

Decomposing $f\left(x_{4}, x_{3}, x_{2}, x_{1}\right)$ into $y\left(x_{4}, x_{3}\right)$ and $z\left(x_{2}, x_{1}\right)$ yields

$$
\begin{aligned}
f\left(x_{4}, x_{3}, x_{2}, x_{1}\right) & =y\left(x_{4}, x_{3}\right) z\left(x_{2}, x_{1}\right) \\
y\left(x_{4}, x_{3}\right) & =x_{3}+x_{4} \\
z\left(x_{2}, x_{1}\right) & =x_{2} x_{1}
\end{aligned}
$$

The fundamental operation on which these restructuring techniques depend (with the exception of simplification) is that of division. The division of one Boolean function by another can be expressed as

$$
\begin{equation*}
F=G Q+R \tag{7.1}
\end{equation*}
$$

where $F$ is a logical expression and $G$ is a Boolean divisor. $Q$ and $R$ are the quotient and remainder, respectively and are logical expressions. If $G$ divides $F$ exactly (without a remainder) then $G$ is a Boolean factor of $F$. The power of Boolean division lies in its use of Boolean identities.

Any logical expression has a large number of subexpressions which are Boolean divisors and factors and in any optimisation algorithm it is, generally, impractical to determine all of these subexpressions. The number of divisors and factors may be reduced by considering only algebraic representations of logic functions. Algebraic representations are defined as being prime and irredundant, that is, no cube of the representation is contained in another cube. An algebraic product of two expressions is valid if, and only if, the expressions have disjoint support. This means that an algebraic product can be obtained by polynomial expansion and makes no use of Boolean identities.
e.g.

$$
\left(x_{1}+x_{3}\right)\left(x_{2}+x_{5} x_{4}\right)=x_{2} x_{1}+x_{5} x_{4} x_{1}+x_{3} x_{2}+x_{5} x_{4} x_{3}
$$

Hence algebraic division can be expressed as

$$
\begin{equation*}
F=G Q+R \tag{7.3}
\end{equation*}
$$

where $G$ is an algebraic divisor of $F$ if, and only if, $Q$, the quotient and $R$, the remainder are algebraic expressions and $Q$ is not null. Algebraic division is less powerful than Boolean division but experiments have shown it to produce good results in a reasonable time-scale [92]. Hence, as is of ten the case for logic synthesis algorithms, there is a trade-off between the quality of results and the time spent realising a solution.

The extensive use of algebraic division in the resynthesis techniques necessitates that a good divisor be identified and that the division operation is executed as efficiently as possible. Candidate divisors are identified from the set of kernels of the algebraic expression. A kernel, $k$, of an algebraic expression, $F$, is the quotient of $F$ and a cube $c$, such that

$$
\begin{equation*}
k=\frac{F}{c} \tag{40}
\end{equation*}
$$

The kernel must be cube-free, that is, it cannot be algebraically factored by any cube. The cube, $c$, used to find kernel, $k$, is termed the co-kernel of $k$. A kernel that contains no kernels other than itself is a level-0 kernel. Generally, a kernel is of level-n if the highest level kernel it contains is a level-( $n-1$ ) kernel. Several methods have been developed for extracting the kernels of an expression, for example, constructing the cube-literal matrix and proceeding to determine prime rectangles [91]. It is possible to extract all kernels of an expression or limit the set to only kernels of a certain level. Again there is a trade-off between quality of the result and time spent executing the algorithm. Having extracted the set of kernels of an expression it is then necessary to select a suitable divisor from this set. The selection of a good divisor generally involves a heuristic as it is impractical to perform all divisions in order to determine the optimum solution.

As previously mentioned Boolean division differs from the weaker algebraic division in that it makes use of Boolean relationships, e.g.

$$
x \cdot \bar{x}=0 \quad x+0=x
$$

The objective of Boolean division is to determine a quotient, $Q$, and
remainder, $R$, where $Q$ and $R$ are as simple as possible. This involves the generation of the 'don't care' set for the expression and the utilisation of these 'don't care' terms to minimise the expression. The goal of minimising $Q$ and $R$ relies on heuristic methods.

The second step in the process of multi-level logic synthesis is to optimise individual nodes in the network. The restructuring techniques so far described have not taken into account the 'don't care' terms of the multilevel network, and it is in the minimisation of individual nodes that 'don't care' terms are utilised. The 'don't care' terms of a multi-level circuit can be classified as external and internal 'don't cares'. External 'don't care' terms are primary input patterns which will never occur for a particular primary output. Internal 'don't care' terms are dependent on the structure of the multi-level circuit. The 'don't care' terms for each network node can be determined, then utilised by a two-level minimiser to simplify the structure of network nodes. Additionally, nodes with small fan-out may be collapsed to create larger nodes. This leads to changes in the structure of the network and may lead to further restructuring. Hence, multi-level logic optimisation is often an iterative process.

The theory of Boolean and algebraic approaches to logic synthesis has been briefly detailed. The development of multi-level synthesis techniques and the synthesis systems in which they are incorporated are now reviewed.

Brayton and McMullen [93] described an algebraic technique for determining subexpressions common to two or more Boolean functions. The method is comprised of several algorithms which identify then extract common multicube expressions before extracting kernels using algebraic division. The algorithm can compute the full set of kernels or employ a heuristic whereby kernel extraction is limited to determining level-0 and level-1 kernels. It is stated that the heuristic method produces results comparable with those obtained for full kernel extraction. If kernel extraction is limited then the additional step of collapsing is applied. This method of subexpression extraction can also be applied to single Boolean expressions to form canonical factorised forms. Additionally, a form of Boolean division was defined which may be applied in place of algebraic division to improve the quality of results at the expense of execution time. The result of
applying the decomposition and factorisation algorithms were illustrated by example. Brayton and McMullen [92] extended their work on multi-level logic synthesis based on the operations of extraction, collapsing, simplification and decomposition. The simplification step employs techniques adapted from the two-level PLA minimiser Espresso-II [63], and uses the 'don't care' terms of the Boolean network. Simplification using 'don't care' terms was defined as Boolean substitution and it was stated that this is more powerful than the algebraic substitution performed in the extraction operation. An example was given to illustrate the performance of the synthesis system formed from the operations mentioned previously.

Techniques for factoring Boolean function were developed by Brayton [94]. The methods are heuristic and provide the opportunity to derive results of varying quality with the penalty of increased execution time for reduced circuit area. Boolean expressions are factored using algebraic and Boolean techniques, and procedures for both methods were described. The method of Boolean division utilised the 'don't care' set of the network and is a heuristic technique. Additionally, the problems of optimal algebraic factorisation in conjunction with the rectangle-covering problem were addressed. The rectangle-covering problem was further considered by Brayton, Rudell, Sangiovanni-Vincentelli and Wang [95]. An algorithm was presented which heuristically determines good rectangle-coverings of the cube-literal matrix of Boolean functions. Additionally, the rectangle-covering problem was applied to factoring and common cube extraction of multi-level networks.

The techniques and algorithms developed in [92, 93, 94, 95, 96] have been amalgamated to form the multi-level optimisation system MIS [97]. MIS adopts a 'global' optimisation approach, first restructuring the network before applying local optimisation techniques to individual nodes. In addition to performing area optimisation the system performs timing optimisation, restructuring logic and trading area for speed. This work illustrates the practical use of factorisation and decomposition techniques for multi-level circuit representations and concludes by presenting results for circuit optimisation.

BOLD [99]. Socrates is a rule-based expert system which performs area and speed optimisation of Boolean functions. The system utilises weak division to decompose the function into a multi-level structure. Espresso-II [63] then utilises the multi-level 'don't care' terms to determine a minimal set of prime implicants for subexpressions of the Boolean network. The Boulder Optimal Logic Design System - BOLD is based on the optimisation algorithms of Espresso-II but additionally employs Boolean resubstitution to produce a multi-level logic representation which is prime and irredundant. Results were given for benchmark functions, and illustrate the performance of BOLD compared with two other multi-level synthesis systems. In addition to producing a multi-level circuit representation BOLD returns the tests for the network, which is $100 \%$ testable for single input stuck faults. Additionally, Bergamaschi [100] introduced SKOL, a logic synthesis and technology mapping system. The algebraic and Boolean techniques for multilevel minimisation are similar to the methods employed by MIS. The original aspects of the package lie in its technology mapping strategy.

It is appropriate to mention some other areas of research into multi-level optimisation. Ykman-Couvreur [101] introduced Phifact, a multi-level optimisation system which employs disjoint Boolean division for incompletely specified Boolean functions to minimise multi-level structures. Karpovsky [102] considered the minimisation of multi-level circuits and implementation as gate arrays. The concept of multi-level prime implicants was presented by Lawler [103]. Generating multi-level prime implicants of a function can determine the absolute minimum form of the function but requires an iterative approach. The importance of the 'don't care' terms in determining optimum multi-level circuits was discussed by Brayton [104]. Brayton, Sentovich and Somenzi [105] introduced global flow analysis, utilising multilevel 'don't care' terms, whilst Bartlett, Brayton, Hachtel et al [106] adapt the concepts of prime and irredundant forms, familiar in two-level minimisation, to multi-level synthesis.

The concept of Reed-Muller multi-level circuits was introduced by Green and Foulk [107] and further considered by Green and Edkins [13]. They described synthesis procedures which express generalised RM expansions of switching functions as multi-level circuits comprised of 2 -input gates. The techniques derive Adaptive Logic Trees (ALTs) where the problem of
deriving a minimal tree can be solved by selecting the appropriate variables at each level of the tree. Two strategies are described for determining the order of selection of input variables. Additionally, the method is extended to include non-binary functions [13].

Saul [85, 108] employed algebraic techniques to develop a logic synthesis system for RM expressions. He introduced the RM factored form, where the OR operator of the traditional Boolean factored form is replaced by the Exclusive-OR operator. As algebraic techniques do not utilise Boolean identities, Saul directly employed the operations of extraction, collapsing, substitution, factoring and decomposition, together with algebraic division, for restructuring RM expressions. One aspect of logic restructuring where the Boolean and RM approaches differ is in extracting common kernels. Saul proved that whilst it is possible to extract 'overlapping' kernels from a Boolean expression, this is not a valid operation in RM multi-level logic synthesis. Node simplification in the RM domain is performed by a two-level minimiser which optimises mixed polarity RM expansions [85]. Results were presented for benchmark functions which illustrate the performance of the multi-level minimiser compared with the two-level RM minimiser. Additionally, Pearce, Saul and Lester [18] described the use of the multi-level synthesis techniques detailed previously $[85,108]$ as a means of implementing logic functions using FPGAs. The performance of this tool was illustrated for a selection of benchmark functions, where the number of cells of the FPGA required to represent the multi-level ESOP form was compared with that required for the traditional multi-level SOP form.

Reed-Muller minimisation techniques were integrated into the Gatemap synthesis system [76] and the Diades design automation system [109]. Gatemap expresses logic functions in both Boolean and mixed polarity RM form, hence the system may choose the 'best' realisation at each stage of the resynthesis. Diades employs a similar strategy, utilising both fixed and mixed polarity RM expansions.

The structure of ROBDDs has been described in section 2.6 of chapter 2. The remainder of this section is dedicated to reviewing methods for deriving ROBDDs representing Boolean functions. Techniques for deriving efficient ROBDDs representations through variable ordering are of
particular interest. Additionally, the graphical representation of ESOP forms using structures analogous to ROBDDs are considered.

The concept of Binary Decision Diagrams (BDDs) was first introduced by Lee [29] who utilised Binary Decision Programs to implement Shannon's theorem. Binary Decision Programs provide an alternative to the conventional algebraic representation of switching functions. Akers [30] extended this work by demonstrating the use of BDDs to graphically represent switching functions and, hence, both combinational and sequential logic functions. Additionally, it was demonstrated that any BDD represents more than one function, that is, by entering the BDD at various points between the root and leaf nodes it is possible to determine the structure of a number of subfunctions. This aspect of BDDs can lead to a reduction in the number of nodes in a BDD through subgraph sharing, and is important when constructing multi-output BDDs. Akers illustrated how the BDD may be used to determine the output of a function for any given input by simply tracing a path through the tree, guided by the condition of the input variables. The idea of reducing the number of nodes in a BDD through node deletion and merging was introduced. This leads to reduced BDDs and it was emphasised that a reduced BDD represents the essential implicants of a Boolean function. It cannot, however, be guaranteed that the BDD represents the essential prime implicants. Thus a reduced BDD may not give the absolute minimum realisation of a function. A method for determining the number of terms in a SOP (POS) form without generating the terms is given, and it is stated that it is possible to employ a similar technique to determine the number of literals in the representation. Additionally, the use of the BDDs as an instrument for generating test sets of combinational logic circuits was discussed. Akers then proceeded to derive an actual physical implementation of a BDD using ' 1 out of 2 selectors', or single control variable multiplexers, with a particular regard to testability. Extracting a circuit from a BDD using path tracing results in a many-gate two-level realisation whilst directly implementing nodes generates a multi-level circuit comprising of fewer gates. The use of inverters in BDDs as a means of simplification was illustrated by example and the concept of allowing nodes to represent subfunction as opposed to variables was introduced. Akers illustrated these techniques by deriving a BDD for an adder circuit.

Bryant [31] further developed the use of BDDs by introducing the concept of ordered BDDs. The OBDD restricts the order in which the input variable may be considered, hence the number of OBDDs which represent a function is somewhat less than the total number of BDDs which can be constructed. Bryant conjectured that 'most commonly encountered functions have a reasonable representation' in OBDD form. Reduced OBDDs, determined using the merger and deletion rules are formally defined, and it was proved that a ROBDD gives a canonical representation of a Boolean function. Bryant recognised that the ordering of variables was very important when constructing minimal BDDs and additionally noted that some functions cannot be efficiently represented using ROBDDs regardless of variable ordering. He termed these 'inherently complex functions' citing the integer multiplier as an example. Bryant detailed a symbolic manipulation program which constructed ROBDDs of functions. The program comprised of subprograms each implementing an algorithm and the time complexity of each of these algorithms was given. Bryant stated that experimental results for the algorithms had proved to be favourable. However, results were dependent on the user choosing a good ordering of the input variables. Finally, Bryant considered the problems of verifying the design of an ALU. The circuit was represented and evaluated for different variable orderings.

Matsunaga and Fujita [110] presented a multi-level optimisation system which uses ROBDDs to represent logic functions. The system was based on the transduction method which optimises by repeatedly transforming and reducing functions. The transduction method utilises permissible functions, which are analogous to 'don't care' terms, and this requires that the ROBDD representation be extended. This was done by introducing a new vertex type, enabling logic functions and permissible functions to be calculated using ROBDDs. Experimental results were given for benchmark functions where the initial implementation is a multi-level circuit representation.

An area in which there has been a substantial amount of research is that of determining a optimum (sub-optimum) order of input variables for ROBDDS. Friedman and Supowit [32] described a technique for determining optimal orderings with the goal of reducing the number of nodes in the network. The method has a lower time complexity than the previous best method and was stated as being most suitable for functions of 11 or fewer
variables. The variable ordering methods presented by Fujita, Matsunaga and Kakuda [111] are based on the technique of exchanging a variable with its neighbour in the ROBDD. Two algorithms were presented, the first is suited to two-level circuits (SOP forms) and heuristically determines an initial variable ordering based on determining the most binate variables in the SOP representation. The second algorithm is suitable for multi-level representations and initial variable orderings are determined heuristically. Both algorithms then proceed to optimise the ROBDD using exchange of variables. The criterion for minimisation is to reduce the number of branch crossings on the ROBDD. Additionally, results were given for benchmark functions. Ishiura, Sawada and Yajima [112] presented exact and heuristic optimisation algorithms for determining minimal OBDD representations. The methods are based on exchanges of variables, with the depth of exchange of variable being increased, thus differing from the technique presented in [111]. Once again, benchmark circuits were tested and results presented.

Besson, Bouzouzou, Floricica, Saucier and Roane [113] computed the set of kernels of a Boolean function to determine good variable orderings for ROBDDs. The algorithm analyses the support of kernels of a Boolean function and derives good variable orderings based on the results of this analysis. Results were presented for benchmark functions which illustrated the performance of the algorithm when compared with other optimisation methods.

Besson, Bouzouzou, Crastes, Floricica and Saucier [114] presented synthesis method for speed and area optimisation. BDDs without variable ordering restrictions were used for speed optimisation, with the goal of selecting the variable ordering which resulted in the terminal nodes being reached as quickly as possible. The method is based on determining the variable occurrences in the SOP form of the Boolean function. Alternatively, area optimisation is performed by determining a 'good' variable ordering for ROBDDS. Three methods for area optimisation were presented, two based on kernel extraction, the third on variable occurrences. The direct relation between a node of a BDD and a 2-to-1 multiplexer is exploited in order to develop a synthesis system suitable for FPGA realisations. Experimental results were presented for benchmark functions and the algorithms are evaluated against existing techniques.

Liaw and Lin [115] presented an analysis of OBDDs for general Boolean functions. They considered the effectiveness of the merger and deletion rules and investigated the sensitivity of Boolean functions to variable ordering. Bounds for worst case sizes of ROBDDs are calculated and the authors conclude that the merger rule is significantly more effective than the deletion rule. Additionally, they stated that a large proportion of general Boolean functions are not sensitive to variable orderings. The work presented is this paper would seem to contradict much of the experimental and theoretical results given in $[31,32,110,111,112,113,114]$. However, it is noted that these conclusions are based on a worst case analysis and do not devalue the use of BDDs.

It is appropriate to mention other aspects of research into BDD utilisation. Matos and Oldfield [116] presented work on the physical implementation of a BDD as a custom or semi-custom device with the structure of an array. They exploited the similarities between the node of a BDD and 2-to-1 multiplexer. Additionally, Abadir and Reghbati [117] presented work on functional test generation from BDD structures. Coudert and Madre [118] exploit the structural properties of BDDs to generate prime and essential prime implicants of Boolean functions.

Purwar [47] introduced the concept of utilising the BDD structure to derive fixed polarity RM expansions of Boolean functions. A BDD represents the essential implicants of a Boolean function, hence no minterm is represented by more than one path of the BDD. By tracing all paths which terminate in a node with the value 1 and applying a set of rules, it is possible to deduce the RM coefficients of any fixed polarity RM expansion. If the same procedure is applied to all paths terminating in a node with the value 0 then the complement of the FPRM expansion is formed, and by using the identity $\bar{x} \oplus 1=x$, it is possible to derive the expansion from its complement. This is useful as it is possible to limit the number of paths which must be traced to determine the $R M$ expansion of the Boolean function. Hence, if a Boolean function has many minterms it may be more efficient to derive the complement then transform this expansion to its true state.
presented using Functional Decision Diagrams (FDDs). These structures, devised by Kebschull, Schubert and Rosenstiel [48], are analogous to BDDs and provide a means of representing any FPRM expansion. Initially, a binary tree is formed which represents a FPRM expansion, the merger and deletion rules for reducing BDDs may then be applied until the number of nodes in the structure can no longer be reduced. The final structure is a FDD which represents the initial FPRM expansion both as a two-level ESOP expression and as a factored form. It is interesting to note that the number of paths through the FDD is reduced through merger and deletion. It is suggested that as this is equivalent to the number of terms in the FPRM expansion then a more efficient representation of the original expansion may be been derived. Results presented compare the numbers of nodes and paths in BDDs and FDDs used to represent initial Boolean functions. These illustrate that, generally, for any Boolean function the number of nodes in a FDD is greater than the number of nodes in the equivalent BDD. This result is reversed when the number of paths through each structure is considered. The number of nodes in any FDD and also the number path through the structure is dependent on the variable ordering of the nodes. Thus, Kebschull et al presented a heuristic algorithm which may be employed to derive minimal (sub-minimal) FDDs. The technique determines the variable ordering by counting the number of occurrences of each of the variables in the FPRM expansion. The variable which occurs most of ten is selected and is represented by the node closest to the root of the FDD. This process is repeated and variables with the highest numbers of occurrences are represented by nodes at levels closest to the root of the FDD. Thus, the FDD grows from root to leaves. A practical application of FDDs was presented by Schubert, Kebschull and Rosenstiel [49]. A technology mapping method was described where a Boolean function is represented in the functional domain by a FDD. Groups of nodes of the FDD may be mapped to configurable logic blocks (CLBs) of a FPGA. The goal of this algorithm is to determine an efficient FPGA implementation of a FDD through grouping nodes.

An alternative means of representing KRO expansions was presented by Sarabi, Ho, Iravani, Daasch and Perkowski [68]. KFDDs may be considered to be an extension to FDDs as any node of a KFDD may represent an expansion variable in either fixed or mixed polarity form. Thus, FDDs and

BDDs are necessarily a subset of KFDDs. An algorithm for determining the optimal (sub-optimal) variable ordering for OKFDDs was presented. This technique employs a similar strategy to that utilised for FDDs. The number of occurrences of the expansion variable of the KRO expansion is determined and, additionally, the polarity of all variables is considered. Sarabi et al presented results which, for a collection of benchmark functions, illustrated the slight benefits of employing OKFDDs as opposed to BDDs and FDDs. Comparisons were made on the basis of the number of nodes in each type of structure. The concept of graphically representing ESOP forms was also been explored by Sasao [7]. Ternary Decision Diagrams [119] may be employed to represent PSDKRO expansions. Indeed. a multilevel representation is ideally suited to the structure of this type of expression.

The following sections of this chapter introduce RMBDTs and RMBDDs, graphical means of representing FPRM expansions. It will become apparent that similarities exist between the structure and application of these RMBDDs and the FDDs examined in the preceding literature review. Therefore, it is necessary to state that RMBDTs and RMBDDs, together with rules for manipulating these structures, were developed independently of FDDs. These two types of graphical representation are not identical. It will be demonstrated that RMBDDs are particulary well suited to representing FPRM expansions which are to be implemented as RM-ULM networks.

### 7.2 Reed-Muller Binary Decision Trees and Diagrams

A Reed-Muller Binary Decision Tree (RMBDT) is a graphical representation of a FPRM expansion and is similar in structure to a BDD used to represent a Boolean function. The RMBDT is a directed acyclic graph comprised of nodes interconnected by branches. Two types of nodes are employed in the representation, namely, terminal nodes and non-terminal nodes. A terminal node (represented by a box) may assume the binary value 0 or 1 and has a single output branch. A non-terminal node (represented by a circle) is associated with a variable $\dot{x}_{j}$ of a FPRM expansion. Each non-terminal node has one output branch and two input branches. The input branch which is denoted 0 ( 0 -input branch) indicates that the node variable is absent, whilst the input branch which is denoted 1 (1-input branch) indicates the presence of the node variable $\dot{x}_{j}$. The state of the node variable, that is,
whether it is present in true or complemented form is dependent on the polarity of the FPRM expansion being represented.

The relationship between a single non-terminal node of a RMBDT and the FPRM expansion being represented by that RMBDT is illustrated in Figure 7.1. The node variable $\dot{x}_{j}(j=1,2, \ldots, n)$ may be equated with the splitting variable in both of the modified forms of the Shannon expansion theorem (section 2.4 chapter 2 ). The expansion

$$
\begin{align*}
f\left(x_{n}, x_{n-1}, \ldots, x_{1}\right)= & f\left(x_{n} x_{n-1}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{2}, x_{1}\right) \\
& \bullet x_{j}\left[f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{2} x_{1}\right) \oplus f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{2}, x_{1}\right)\right] \tag{7.1}
\end{align*}
$$

with literal $x_{j}$ as the splitting variable is represented by the non-terminal node represented in Figure 7.1, where $X=0$.
The non-terminal node which represents the expansion

$$
\begin{align*}
f\left(x_{n}, x_{n-1}, \ldots, x_{1}\right)= & f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{2}, x_{1}\right) \\
& \oplus \bar{x}_{j}\left[f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{2} x_{1}\right) \oplus f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{2}, x_{1}\right)\right] \tag{7.2}
\end{align*}
$$

with literal $\bar{x}_{j}$ as the splitting variable is also illustrated in Figure 7.1, however in this instance, $X=1$.

$$
\begin{aligned}
& f\left(x_{n}, x_{n-1}, \ldots, x_{1}\right) \\
& f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, x_{1}, x_{j-1}, \ldots, x_{1}\right) \quad\left(f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{1}\right) \oplus\right. \\
& \left.f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{1}\right)\right)
\end{aligned}
$$

Figure 7.1: Non-terminal node representing function variable $x_{j}$

The subfunctions of ( $n-1$ ) variables

$$
\begin{equation*}
f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{2} x_{1}\right) \tag{7.3}
\end{equation*}
$$

$$
\begin{equation*}
f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{2}, x_{1}\right) \tag{7.4}
\end{equation*}
$$

$$
\begin{equation*}
f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{2} x_{1}\right) \oplus f\left(x_{n} x_{n-1}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{2}, x_{1}\right) \tag{7.5}
\end{equation*}
$$

are independent of variable $x_{j}$ and are subfunctions of the original FPRM expansion. Indeed, they are FPRM expansions of ( $n-1$ ) variables.

The $n$ variable FPRM expansion is

$$
\begin{equation*}
f\left(x_{n}, x_{n-1}, \ldots, x_{1}\right)=b_{0} \oplus b_{1} \dot{x}_{1} \oplus b_{2} \dot{x}_{2} \oplus b_{3} \dot{x}_{2} \dot{x}_{1} \oplus \ldots \ldots \oplus b_{2^{n-1}} \dot{x}_{n} \dot{x}_{n-1} \ldots \dot{x}_{2} \dot{x}_{1} \tag{7.6}
\end{equation*}
$$

where $b_{i} \in\{0,1\}\left(i=0,1, \ldots \ldots, 2^{n}-1\right)$ and $\dot{x}_{j}=x_{j}$ or $\bar{x}_{j}(j=1,2, \ldots, n)$ This expansion may be represented by connecting together terminal and non-terminal nodes to form a RMBDT. One possible structure is illustrated in Figure 7.2 where the order of the expansion variables from the lowest level of non-terminal nodes to the root of the RMBDT is $\langle 1,2, \ldots, n-1, n\rangle$.

Hence, any FPRM expansion may be represented by a RMBDT comprised of $n$ levels of non-terminal nodes and a single level of terminal nodes, all interconnected by branches. The terminal nodes assume the values of the coefficients $b_{i}$ of the FPRM expansion. The RMBDT is, therefore, a functional domain representation of a switching function. This may be contrasted with the BDD of section 2.6 of chapter 2, where the terminal nodes assume the values of the coefficients $d_{i}$ of a Boolean function. A BDD is an operational domain representation of a switching function.

The path from the root of the RMBDT to each terminal node with the value $b_{1}$ covers $n$ nodes and represents the $i$ th product term of the FPRM expansion. If the path is via the 0 -input branch of a node then the node
variable is absent from the product term. If the path is via the 1 -input branch of a node then the node variable is present in the product term.


Figure 7.2: Ordered Reed-Muller Binary Decision Tree of $n$ variable FPRM expansion.

The following example illustrates the use of a RMBDT to represent a 4 variable FPRM expansion.

Example 7.1 Construct the RMBDT of the following 4 variable polarity 0 FPRM expansion.
$n=4, p=0$,

$$
\begin{equation*}
f_{0}\left(x_{4}, x_{3}, x_{2}, x_{1}\right)=1 \oplus x_{1} \oplus x_{2} x_{1} \oplus x_{3} \oplus x_{4} \oplus x_{4} x_{1} \oplus x_{4} x_{2} x_{1} \oplus x_{4} x_{3} x_{1} \oplus x_{4} x_{3} x_{2} x_{1} \tag{7.7}
\end{equation*}
$$

The RMBDT is illustrated in Figure 7.3. The order of the variables from the lowest level of non-terminal nodes to the root of the RMBDT is $<1,2,3,4>$.


Figure 7.3: ORMBDT of a 4-variable RM expansion (polarity 0), Example 7.1
The RMBDTs which have been described are ordered structures, and henceforth, will be denoted as ordered RMBDTs. They may be defined in a manner similar to that given for OBDTs (section 2.6 chapter 2). The 0 -input and 1 -input branches of the root of the ORMBDT are connected to the output branches of the 2 nodes present at level 2. In general, at level $\ell$, there are $2^{\ell-1}$ nodes and all have the same node variable, $\dot{x}_{k}$. Each output branch of the $2^{\ell-1}$ nodes is connected to the input branches of the nodes at level $(\ell-1)$ and the $2^{\ell}$ input branches are connected to the output branches of the $2^{\ell}$ nodes at level $(\ell+1)$. At the final level, that is level ( $n+1$ ), there are $2^{n}$ terminal nodes and the output branch of each is connected to one of the input branches of the nodes at level $n$. The ORMBDT of any $n$ variable FPRM expansion is comprised of $2^{n+1}-1$ nodes, of which $2^{n}-1$ are non-terminal nodes and $2^{n}$ are terminal nodes.

The procedure for constructing ORMBDTs relies on two basic equations which are derived from the Shannon expansion theorem [26, 68]. The FPRM expression should be expanded about any expansion variable. Equation (7.1) should be employed if the variable is present in true form throughout the GRM expansion. Alternatively, Equation (7.2) should be employed if the variable is present in complemented form throughout the FPRM expansion.

This results in two subexpansions each of which may be expanded about another expansion variable, employing the appropriate equation. This should be repeated until the FPRM expression has been expanded about each variable. The ORMBDT may be constructed as the FPRM expression is being expanded. It is also possible to employ this procedure to construct an ORMBDT from a switching function expressed in Boolean SOP or disjoint SOP form. The choice of the equations employed to expand the function about each variable is dictated by the state of that variable in both the ORMBDT and the final FPRM expansion.

The number of nodes in an OBDT representing any Boolean function may be reduced through merger and deletion to form a ROBDD. A similar reduction procedure may be applied to an ORMBDT to form an ORMBDD. In order to reduce the number of nodes in an ORMBDT is it necessary to identify both equivalent and redundant nodes.

## Equivalent nodes

Two terminal nodes of an ORMBDT are equivalent if they each have the same Boolean value.

Two non-terminal nodes of an ORMBDT are equivalent if both nodes represent the same expansion variable, the subtrees rooted at the 0 input branches of these nodes are identical, and subtrees rooted at the 1 -input branches are identical.

## Redundant nodes

A non-terminal node of an ORMBDT is redundant if the 1 -input branch is connected to a terminal node with the value 0 .

The definitions of equivalent and redundant nodes leads to the formulation of two rules which may be employed to reduce the number of nodes in an ORMBDT.

## Merger rule

If two terminal nodes, $a$ and $b$, of an ORMBDT are equivalent then redirect the input branch connected to the output branch of node $b$ to the output branch of node $a$. Node $b$ can then be deleted.
If two non-terminal nodes, $a$ and $b$, of an ORMBDT are equivalent
then redirect the input branch connected to the output branch of node $b$ to the output branch of node $a$. Delete node $b$ and its subtree.

## Deletion rule

If a non-terminal node is redundant then redirect the input branch connected to the output branch of the redundant node to the subtree rooted at the 0 -input branch of the node. Delete the node.

The merger and deletion rules are illustrated in Figure 7.4 and Figure 7.5, respectively.


Figure 7.4: Merger rule

The ORMBDD is a reduced form of the ORMBDT where the output branch of any node in the ORMBDD may be connected to the input branches of more than one node. If the reduction rules are repeatedly applied to an ORMBDT until the number of nodes in the structure can no longer be reduced then a reduced ORMBDD is formed. The RORMBDD comprises of a minimum number of nodes for a given variable ordering, and is a unique representation of the initial FPRM expansion. A RORMBDD can, therefore, be defined as a canonical representation of a FPRM expansion.


Figure 7.5: Deletion rule

The following example illustrates the use of the merger and deletion rules.

Example 7.2 The ORMBDT illustrated in Figure 7.3 may be reduced using the merger and deletion rules. The resulting RORMBDD is shown in Figure 7.6.


Figure 7.6: RORMBDD of Example 7.2

A ROBDD with a fixed variable ordering is a canonical representation of a Boolean function [31]. This is also true for RORMBDDs i.e. every FPRM expansion has a unique representation. Thus, for a given FPRM expansion and a fixed variable ordering, the RORMBDD has a minimum number of nodes and no further reduction is possible. It is, however, possible to reduce the number of paths through a RORMBDD by complementing the state of certain decision variables. If the two input branches of any level $\ell$ node are both connected to the output branch of the same level $(\ell+1)$ node then disconnect the 0 -input branch and attach it to a leaf with the value 0 . Complement the state of the level $\ell$ node variable. The resulting structure will no longer represent a FPRM expansion but instead represents an ESOP form. This is illustrated in Figure 7.7, where the RORMBDD includes decision variables which are present in complemented form. The initial RORMBDD representing the FPRM expansion of Equation (7.7) is given in Figure 7.6 where the structure represents 9 product terms. The number of paths through the RORMBDD may be reduced to 6 as illustrated in Figure 7.7. This new structure represents the ESOP expression of Equation (7.8).

$$
\begin{equation*}
f\left(x_{4}, x_{3}, x_{2}, x_{1}\right)=\bar{x}_{1} \oplus x_{2} x_{1} \oplus x_{3} \oplus x_{4} \bar{x}_{1} \oplus x_{4} x_{2} x_{1} \oplus x_{4} x_{3} \bar{x}_{2} x_{1} \tag{7.8}
\end{equation*}
$$

ORMBDTs and ORMBDDs may be derived by applying the modified forms of the Shannon expansion theorem (Equations (7.1) and (7.2)) to the FPRM expansion. The structures may then be reduced to a RORMBDD by applying the merger and deletion rules.

Observing the similarities between ROBDDs and RORMBDDs, it is possible to make two further comments on the application of RORMBDDs. It is suggested that RORMBDDs may be used to represent multi-output FPRM expansions and ESOP forms and that any non-terminal node of this graphical structure may be employed to represent a subfunction of the FPRM expansion or ESOP form.

### 7.3 Minimising RORMBDD Representations through Variable Ordering

 The number of nodes in the RORMBDD of a FPRM expansion, for a particular order of variables, is a minimum and cannot be reduced. However, it has

Figure 7.7: RORMBDD of the ESOP expansion.
been observed that the number of nodes in a ROBDD representing a Boolean function is affected by the order in which the function variables are associated with the nodes of the structure [31]. Due to the similarities between ROBDDs and RORMBDDs, it is possible to state that the number of nodes in a RORMBDD may be altered by adjusting the order in which expansion variables are assigned to nodes. It is possible to construct $n$ ! RORMBDDs of any FPRM expansion of $n$ variables, where each structure employs a different variable ordering. As the number of expansion variables increases the number of possible RORMBDDs grows rapidly. Hence, it is desirable to develop efficient algorithms which determine optimum (suboptimum) variable orderings and allow the construction of RORMBDDs which efficiently represent FPRM expansions. A heuristic algorithm is now presented which determines optimum (sub-optimum) variable orderings for RORMBDDs of FPRM expansions. The algorithm is not exhaustive and employs simple, if somewhat crude, techniques drawn from the approaches detailed in $[48,68,114]$.

### 7.3.1 Variable Ordering Algorithm for Deriving RORMBDDs

The algorithm presented in this section determines the order of variables, from root to leaves, of an ORMBDD. The overall structure of this heuristic technique is now described briefly as a precursor to the fuller, stepwise form presented in the latter part of this section. The FPRM expansion which is to be represented by an ORMBDD is initially inspected to determine whether it is comprised of two identical subfunctions. If this is the case then merger is possible and the variable about which the identical subfunctions may be formed is selected as the node variable at this level. Otherwise, the variable which appears most often in the FPRM expansion is chosen as the node variable. The FPRM expansion can then be divided into 2 subfunctions. The inspection is then repeated using the new subfunctions and a single node variable is selected at each level. The process is repeated until all variables are allocated.

FPRM expansions may be expanded using two basic expressions derived from the Shannon expansion theorem. The expressions were derived in section 7.2 and are now repeated.

$$
\begin{aligned}
f\left(x_{n}, x_{n-1}, \ldots, x_{1}\right)= & f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{2}, x_{1}\right) \\
& \oplus x_{j}\left[f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{2} x_{1}\right) \oplus f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{2}, x_{1}\right)\right] \\
f\left(x_{n}, x_{n-1}, \ldots, x_{1}\right)= & f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{2}, x_{1}\right) \\
& \oplus \bar{x}_{j}\left[f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{2} x_{1}\right) \oplus f\left(x_{n}, x_{n-1}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{2}, x_{1}\right)\right]
\end{aligned}
$$

Thus, any FPRM expansion may be expressed as a modulo-2 sum of 2 subfunctions of ( $n-1$ ) variables. The first subfunction $f\left(x_{n}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{1}\right)$ $\left(f\left(x_{n}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{1}\right)\right)$ is independent of literal $x_{j}\left(\bar{x}_{j}\right)$. The second subfunction $f\left(x_{n}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{1}\right) \oplus f\left(x_{n}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{1}\right)$, although independent of the expansion variable, can be said to be associated with the literal $\dot{x}_{f}$

Let $\left[x_{n}, x_{n-1}, \ldots, x_{1}\right]$ represent the original FPRM expansion of $n$ variables. Then $\left[x_{n}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{1}\right]$ represents the subfunction which is independent of literal $\dot{x}_{j}$. That is, $\left[x_{n}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{1}\right]$ represents the subfunction
$f\left(x_{n}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{1}\right)$ if the literal $x_{j}$ is present in the original FPRM expansion. If literal $\bar{x}_{j}$ is present in the FPRM expansion then [ $x_{n}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{1}$ ] represents the subfunction $f\left(x_{n}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{1}\right)$. Let $\left[x_{n}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{1}\right]$ represent the subfunction $f\left(x_{n}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{1}\right) \oplus$ $f\left(x_{n}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{1}\right)$ associated with the expansion variable $x_{j}$ of the original FPRM expansion.

This notation may be extended to allow the representation of a FPRM expansion which has been expanded about 2 or more variables, e.g. for expansion about 2 variables, $x_{j}$ and $x_{k}$
[ $x_{n}, \ldots, x_{k+1}, 0, x_{k-1}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{1}$ ] represents the subfunction which is independent of expansion variables $x_{j}$ and $x_{k}$.
[ $x_{n}, \ldots, x_{k+1}, 1, x_{k-1}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{1}$ ] represents the subfunction which is independent of expansion variable $x_{j}$ but associated with variable $x_{k}$. [ $x_{n}, \ldots, x_{k+1}, 0, x_{k-1}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{1}$ ] represents the subfunction which is associated with expansion variable $x_{j}$ but independent of variable $x_{k}$. [ $x_{n}, \ldots, x_{k+1}, 1, x_{k-1}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{1}$ ] represents the subfunction which is associated with expansion variables $x_{j}$ and $x_{k}$.

As each subfunction is derived from an initial FPRM expansion, the subfunctions are also FPRM expansions. Thus, with regard to Figure 7.1, [ $x_{n}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{1}$ ] represents the 0 -input branch of a non-terminal node with node variable $x_{j}$. The 1 -input branch of the node is represented by $\left[x_{n}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{1}\right]$.
Let $P\left[x_{n}, x_{n-1}, \ldots, x_{1}\right]$ denote the number of product terms in the FPRM expansion represented by [ $x_{n}, x_{n-1}, \ldots, x_{1}$ ].
Let $N_{k}\left[x_{n}, x_{n-1}, \ldots, x_{1}\right]$ denote the number of occurrences of literal $\dot{x}_{k}$ in the FPRM expansion represented by $\left[x_{n}, x_{n-1}, \ldots, x_{1}\right]$ ( $k \in\{1,2, \ldots, n\}$ ).
$\operatorname{Adj}_{k}\left[x_{n}, x_{n-1}, \ldots, x_{1}\right]$ denote the number of product terms adjacent in literal $\dot{x}_{k}$, in the FPRM expansion represented by $\left[x_{n}, x_{n-1}, \ldots, x_{1}\right](k \in\{1,2, \ldots, n\})$.
Let $l$ denote the level of the ORMBDD, $\ell=1,2, \ldots, n$, and novars represent the number of expansion variables which have not yet been employed as node variables.
$N_{\max }$ represents the maximum value of $N_{k}\left[x_{n}, x_{n-1}, \ldots, x_{1}\right]$ found from all FPRM expansions.
These definitions may be extended to FPRM expansions of $j$ variables where $j=1,2, \ldots, n$.

## Variable Ordering Algorithm

S. 1 Form the initial $n$ variable FPRM expansion. Represent the expansion as $\left[x_{n}, x_{n-1}, \ldots, x_{1}\right]$. Set $\ell=1$ and novars $=n$. If $l=n$ then go to $S .10$ else go to S.2.
S. 2 Find $P\left[x_{n}, x_{n-1}, \ldots, x_{1}\right]$. For $j=1,2, \ldots, n$ find $N_{j}\left[x_{n}, x_{n-1}, \ldots, x_{1}\right]$.
S. 3 If $\left(P\left[x_{n}, x_{n-1}, \ldots, x_{1}\right] \bmod 2\right)=0$ then go to S.3a, else go to S.3c.
(Note: If the FPRM expansion is comprised of an even number of product terms then it is possible that it may be divided into two identical subfunctions. If, however, the expansion is comprised of an odd number of product terms then it cannot be divided into two identical subfunctions. This is determined by observing the remainder of integer division by 2.)
S.3a If any $N_{j}\left[x_{n}, x_{n-1}, \ldots, x_{1}\right]=P\left[x_{n}, x_{n-1}, \ldots, x_{1}\right] / 2$ then go to $S .3 b$ else go to S.3c.
S.3b For all $j$ where $N_{j}\left[x_{n}, x_{n-1}, \ldots, x_{1}\right]=P\left[x_{n}, x_{n-1}, \ldots, x_{1}\right] / 2$, find $\operatorname{Adj}_{j}\left[x_{n}, x_{n-1}, \ldots, x_{1}\right]$. If any $\operatorname{Adj}_{j}\left[x_{n}, x_{n-1}, \ldots, x_{1}\right]=N_{j}\left[x_{n}, x_{n-1}, \ldots, x_{1}\right]$ then the FPRM expansion may be expanded into two identical subfunctions (merger). Therefore, select $x_{j}$ as the node variable at level 1 , the root of the ORMBDD. If there is more than one variable $x_{j}$ for which $A d j_{j}\left[x_{n}, x_{n-1}, \ldots, x_{1}\right]=N_{j}\left[x_{n}, x_{n-1}\right.$. $\ldots, x_{1}$ ] arbitrarily choose the decision variable $x_{j}$ where $\operatorname{Adj}_{j}\left[x_{n}, x_{n-1}, \ldots, x_{1}\right]=N_{j}\left[x_{n}, x_{n-1}, \ldots, x_{1}\right]$. If, for all $j$ where $N_{j}\left[x_{n}, x_{n-1}, \ldots, x_{1}\right]=P\left[x_{n}, x_{n-1}, \ldots, x_{1}\right] / 2$, $\operatorname{Adj}_{j}\left[x_{n}, x_{n-1}, \ldots, x_{1}\right] \neq N_{j}\left[x_{n}, x_{n-1}, \ldots, x_{1}\right]$ then S.3c.
S.3c Find $N_{\max }$. The node variable is $x_{j}$ where $N_{j}\left[x_{n}, x_{n-1}, \ldots, x_{1}\right]=N_{\max }$ If there is more than one variable $x_{j}$ for which $N_{j}\left[x_{n}, x_{n-1}, \ldots, x_{1}\right]$ $=N_{\max }$, then arbitrarily choose the decision variable $x_{j}$, where $N_{j}\left[x_{n}, x_{n-1}, \ldots, x_{1}\right]=N_{\max }$.
S. 4 Set $\ell=\ell+1$ and novars $=$ novars -1 . If $\ell<n$ then expand the FPRM expansion about variable $x_{j}$ and go to S.5 else go to S.10.
S. 5 Represent each of the $2^{\text {h-1 }}$ FPRM expansion of novars variables by the symbols $\left[x_{n}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{1}\right], \ldots,\left[x_{n}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{1}\right]$, where the variables selected as node variables are replaced by 0 s or 1 s .
S. 6 For each FPRM expansion of novars variables determine $P\left[x_{n}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{1}\right], \ldots, P\left[x_{n}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{1}\right]$.
S. 7 For each FPRM expansion of novars variables and for each expansion variable $x_{k}$, determine $N_{k}\left[x_{n}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{1}\right], \ldots, N_{k}\left[x_{n^{\prime}}, \ldots, x_{j+1}, 1, x_{j-1}\right.$
$\left.\ldots, x_{1}\right], k \in\{1,2, \ldots, n\} \quad k \neq j$.
S. 8 If, for any FPRM expansion of novars variables ( $P\left[x_{n}, \ldots, x_{j+1}, 0, x_{j-1}\right.$. $\left.\left.\ldots, x_{1}\right] \bmod 2=0\right), \ldots,\left(P\left[x_{n}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{1}\right] \bmod 2=0\right)$ then go to S.8a, else go to S.8c.
S.8a Considering only FPRM expansions for which ( $P\left[x_{n}, \ldots, x_{j+1}, 0\right.$, $\left.\left.x_{j-1}, \ldots, x_{1}\right] \bmod 2=0\right), \ldots,\left(P\left[x_{n}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{1}\right] \bmod 2=0\right)$, if any $\left(N_{k}\left[x_{n}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{1}\right]=P\left[x_{n}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{1}\right] / 2\right)$, $\ldots,\left(N_{k}\left[x_{n}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{1}\right]=P\left[x_{n}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{1}\right] / 2\right)$ then go to S .8 b else go to S.8c.
S.8b For the FPRM expansions which satisfy S.8a, find $A d j_{k}\left[x_{n}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{1}\right], \ldots, A d j_{k}\left[x_{n}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{1}\right]$. If, for any FPRM expansion, $\left(A d j_{k}\left[x_{n}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{1}\right]=\right.$ $\left.N_{k}\left[x_{n}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{1}\right]\right) \ldots \ldots,\left(A d j_{k}\left[x_{n}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{1}\right]=\right.$ $\left.N_{k}\left[x_{n}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{1}\right]\right)$ then that FPRM expansion may be expanded into two identical subfunctions (merger). Therefore, select $x_{k}$ as the node variable at level $\ell$ of the ORMBDD. If there is more than one variable $x_{k}$ for which $\left(\operatorname{Adj}_{k}\left[x_{n}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{1}\right]=N_{k}\left[x_{n}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{1}\right]\right)$, $\ldots,\left(\operatorname{Adj}_{k}\left[x_{n}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{1}\right]=N_{k}\left[x_{n}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{1}\right]\right)$ then arbitrarily select the decision variable $x_{k}$ where $\left(\right.$ Adj $\left._{k}\left[x_{n}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{1}\right]=N_{k}\left[x_{n}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{1}\right]\right)$, $\ldots \ldots\left(\operatorname{Adj}_{k}\left[x_{n}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{1}\right]=N_{k}\left[x_{n}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{1}\right]\right)$.
If, for all $k,\left(\operatorname{Adj}_{k}\left[x_{n}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{1}\right] \neq N_{k}\left[x_{n}, \ldots, x_{j+1}, 0, x_{j-1}\right.\right.$. $\left.\left.\ldots, x_{1}\right]\right), \ldots .,\left(\operatorname{Adj}_{k}\left[x_{n}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{1}\right] \neq N_{k}\left[x_{n}, \ldots, x_{j \bullet 1}, 1, x_{j-1}, \ldots, x_{1}\right]\right)$ then S.8c.
S.8c Considering all FPRM expansions of novars variables, find $N_{\text {max }}$ The node variable is $x_{k}$ where $\left(N_{k}\left[x_{n}, \ldots, x_{j+1}, 0, x_{j-1}, \ldots, x_{1}\right]=N_{\max }\right)$, $\ldots,\left(N_{k}\left[x_{n}, \ldots, x_{j+1}, 1, x_{j-1}, \ldots, x_{1}\right]=N_{\max }\right)$. If there is more than one variable which satisfies the criterion of S.8c then arbitrarily choose the decision variable $x_{k}$ from the variables which satisfy this criterion.
S. 9 Set $\ell=\ell+1$, novars = novars - 1 . If $\ell<n$ then expand each FPRM expansion about variable $x_{k}$ and go to S.5, else go to S.10.
S. 10 The remaining variable is the node variable at level $n$. The variable ordering has been determined where the first variable selected is the node variable at level 1 or the root of the ORMBDD.

This algorithm was implemented in Pascal and the programs executed on Hewlett Packard workstations and a Dell P60 personal computer.

The following example illustrates the use of the variable ordering algorithm. The FPRM expansion is represented using the tabular notation (section 3.2.1 of chapter 3 ).

Example 7.3 Represent the polarity 0 FPRM expansion

$$
\begin{aligned}
f_{0}\left(x_{5}, x_{4}, x_{3}, x_{2}, x_{1}\right)= & 1 \oplus x_{2} \oplus x_{2} x_{1} \oplus x_{3} x_{2} \oplus x_{4} x_{1} \oplus x_{4} x_{3} x_{2} x_{1} \oplus x_{3} \oplus x_{3} x_{2} \\
& \oplus x_{3} x_{3} \oplus x_{5} x_{3} x_{2} x_{1} \oplus x_{3} x_{4} x_{2}
\end{aligned}
$$

by a RORMBDD. Employ the variable ordering algorithm to determine a 'good' order of variables.
S. 1 The FPRM expansion may be represented thus $\left[x_{5}, x_{4}, x_{3}, x_{2}, x_{1}\right]$. $\ell=1$ and novars $=5$.
S. 2 Count the number of piterms, $P\left[x_{5}, x_{4}, x_{3}, x_{2}, x_{1}\right]=11$. Count the number of occurrences of each expansion variable (identical to counting the number of ones in each column $x_{j}$ of the tabular representation of the expansion.

$$
\begin{aligned}
& \begin{array}{lllll}
x_{5} & x_{4} & x_{3} & x_{2} & x_{1} \\
\hline 0 & 0 & 0 & 0 & 0
\end{array} \\
& \begin{array}{lllll}
0 & 0 & 0 & 1 & 0
\end{array} \\
& \begin{array}{lllll}
0 & 0 & 0 & 1 & 1
\end{array} \\
& \begin{array}{lllll}
0 & 0 & 1 & 1 & 0
\end{array} \\
& \begin{array}{lllll}
0 & 1 & 0 & 0 & 1 \\
0 & 1 & 1 & 1 & 1
\end{array} \\
& \begin{array}{lllll}
0 & 1 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 & 0
\end{array} \\
& 1000010 \\
& \begin{array}{lllll}
1 & 0 & 1 & 0 & 0
\end{array} \\
& \begin{array}{lllll}
1 & 0 & 1 & 1 & 1
\end{array} \\
& \begin{array}{llllll} 
& 1 & 1 & 0 & 1 & 0 \\
\hline N_{j}\left[x_{5}, x_{4}, x_{3}, x_{2}, x_{1}\right] & 5 & 3 & 4 & 7 & 4
\end{array} \\
& j=1,2,3,4,5 \\
& N_{5}\left[x_{5}, x_{4}, x_{3}, x_{2}, x_{1}\right]=5, \quad N_{4}\left[x_{5}, x_{4}, x_{3}, x_{2}, x_{1}\right]=3, N_{3}\left[x_{5}, x_{4}, x_{3}, x_{2}, x_{1}\right]=4 \text {, } \\
& N_{2}\left[x_{5}, x_{4}, x_{3}, x_{2}, x_{1}\right]=7, N_{1}\left[x_{5}, x_{4}, x_{3}, x_{2}, x_{1}\right]=4 .
\end{aligned}
$$

S.3a $P\left[x_{5}, x_{4}, x_{3}, x_{2}, x_{1}\right] \bmod 2=11 \bmod 2 \neq 0$, therefore no merger. Go to S.3c.
S.3c $\quad N_{\max }=7$.
$N_{2}\left[x_{5}, x_{4}, x_{3}, x_{2}, x_{1}\right]=7=N_{\max }$. Therefore, the node variable at level 1 is $x_{2}$.
S. $4 \quad l=2$, novars $=4$.
$\ell<5$, therefore, expand the FPRM expansion about variable $x_{2}$.
$f\left(x_{5}, x_{4}, x_{3}, 0, x_{1}\right)=1 \oplus x_{4} x_{1} \oplus x_{5} \oplus x_{5} x_{3}$
$f\left(x_{5}, x_{4}, x_{3}, 0, x_{1}\right) \oplus f\left(x_{5}, x_{4}, x_{3}, 1, x_{1}\right)=1 \oplus x_{1} \oplus x_{3} \oplus x_{4} x_{3} x_{1} \oplus x_{5}$ $\oplus x_{5} x_{3} x_{1} \oplus x_{5} x_{4}$
S. $5 \quad\left[x_{5}, x_{4}, x_{3}, 0, x_{1}\right]$ represents the FPRM expansion

$$
1 \oplus x_{4} x_{1} \oplus x_{5} \oplus x_{5} x_{3}
$$

[ $x_{5}, x_{4}, x_{3}, 1, x_{1}$ ] represents the FPRM expansion

$$
1 \oplus x_{1} \oplus x_{3} \oplus x_{4} x_{3} x_{1} \oplus x_{5} \oplus x_{5} x_{3} x_{1} \oplus x_{5} x_{4}
$$

S. 6 Count the number of piterms in each of the new FPRM expansions, $P\left[x_{5}, x_{4}, x_{3}, 0, x_{1}\right]=4$ and $P\left[x_{5}, x_{4}, x_{3}, 1, x_{1}\right]=7$.
S. 7 Count the number of ones in each column of the tabular technique
representation.

S. $8 \quad P\left[x_{5}, x_{4}, x_{3}, 0, x_{1}\right] \bmod 2=4 \bmod 2=0$.
$P\left[x_{5}, x_{4}, x_{3}, 1, x_{1}\right] \bmod 2=7 \bmod 2=1$.
Now, $P\left[x_{5}, x_{4}, x_{3}, 0, x_{1}\right] \bmod 2=0$, therefore go to S.8a.
S.8a $N_{5}\left[x_{5}, x_{4}, x_{3}, 0, x_{1}\right]=2=P\left[x_{5}, x_{4}, x_{3}, 0, x_{1}\right] / 2$.
$N_{4}\left[x_{5}, x_{4}, x_{3}, 0, x_{1}\right]=1 \neq P\left[x_{5}, x_{4}, x_{3}, 0, x_{1}\right] / 2$.
$N_{3}\left[x_{5}, x_{4}, x_{3}, 0, x_{1}\right]=1 \neq P\left[x_{5}, x_{4}, x_{3}, 0, x_{1}\right] / 2$.
$N_{1}\left[x_{5}, x_{4}, x_{3}, 0, x_{1}\right]=1 \neq P\left[x_{5}, x_{4}, x_{3}, 0, x_{1}\right] / 2$.
Now, $N_{5}\left[x_{5}, x_{4}, x_{3}, 0, x_{1}\right]=2=P\left[x_{5}, x_{4}, x_{3}, 0, x_{1}\right] / 2$, therefore go to S.8b.
S.8b $A d j_{5}\left[x_{5}, x_{4}, x_{3}, 0, x_{1}\right]=1 \neq N_{5}\left[x_{5}, x_{4}, x_{3}, 0, x_{1}\right]$, therefore no merger is possible, go to S.8c.
S.8c $\quad N_{\max }=3$.
$N_{5}\left[x_{5}, x_{4}, x_{3}, 1, x_{1}\right]=N_{3}\left[x_{5}, x_{4}, x_{3}, 1, x_{1}\right]=N_{1}\left[x_{5}, x_{4}, x_{3}, 1, x_{1}\right]=3$,
$N_{5}\left[x_{5}, x_{4}, x_{3}, 0, x_{1}\right]=2>\left(N_{4}\left[x_{5}, x_{4}, x_{3}, 0, x_{1}\right], N_{3}\left[x_{5}, x_{4}, x_{3}, 0, x_{1}\right], N_{1}\left[x_{5}, x_{4}, x_{3}, 0, x_{1}\right]\right)$
Therefore, $x_{5}$ is the node variable at level 2.
S. $9 \quad \ell=3$, novars $=3$
$\ell<5$ therefore, expand the FPRM expansions about variable $x_{5}$. $f\left(0, x_{4}, x_{3}, 0, x_{1}\right)=1 \oplus x_{4} x_{1}$

$$
\begin{aligned}
& f\left(0, x_{4}, x_{3}, 0, x_{1}\right) \oplus f\left(1, x_{4}, x_{3}, 0, x_{1}\right)=1 \oplus x_{3} \\
& f\left(0, x_{4}, x_{3}, 1, x_{1}\right)=1 \oplus x_{1} \oplus x_{3} \oplus x_{4} x_{3} x_{1} \\
& f\left(0, x_{4}, x_{3}, 1, x_{1}\right) \oplus f\left(1, x_{4}, x_{3}, 1, x_{1}\right)=1 \oplus x_{3} x_{1} \oplus x_{4}
\end{aligned}
$$

S. $5 \quad\left[0, x_{4}, x_{3}, 0, x_{1}\right]$ represents the FPRM expansion $1 \oplus x_{4} x_{1}$
[ $1, x_{4}, x_{3}, 0, x_{1}$ ] represents the FPRM expansion $1 \oplus x_{3}$
$\left[0, x_{4}, x_{3}, 1, x_{1}\right]$ represents the FPRM expansion $1 \oplus x_{1} \oplus x_{3} \oplus x_{1} x_{3} x_{1}$ [ $1, x_{4}, x_{3}, 1, x_{1}$ ] represents the FPRM expansion $1 \oplus x_{3} x_{1} \oplus x_{4}$
S. 6 Count the number of piterms in each of the new FPRM expansions, $P\left[0, x_{4}, x_{3}, 0, x_{1}\right]=2, P\left[1, x_{4}, x_{3}, 0, x_{1}\right]=2, P\left[0, x_{4}, x_{3}, 1, x_{1}\right]=4, P\left[1, x_{4}, x_{3}, 1, x_{1}\right]$ $=3$.
S. 7 Count the number of ones in each column of the tabular technique representation.

$$
\begin{aligned}
& N_{4}\left[0, x_{4}, x_{3}, 0, x_{1}\right]=1, N_{3}\left[0, x_{4}, x_{3}, 0, x_{1}\right]=0, N_{1}\left[0, x_{4}, x_{3}, 0, x_{1}\right]=1, \\
& N_{4}\left[1, x_{4}, x_{3}, 0, x_{1}\right]=0, N_{3}\left[1, x_{4}, x_{3}, 0, x_{1}\right]=1, N_{1}\left[1, x_{4}, x_{3}, 0, x_{1}\right]=0, \\
& N_{4}\left[0, x_{4}, x_{3}, 1, x_{1}\right]=1, N_{3}\left[0, x_{4}, x_{3}, 1, x_{1}\right]=2, N_{1}\left[0, x_{4}, x_{3}, 1, x_{1}\right]=2, \\
& N_{4}\left[1, x_{4}, x_{3}, 1, x_{1}\right]=1, N_{3}\left[1, x_{4}, x_{3}, 1, x_{1}\right]=1, N_{2}\left[1, x_{4}, x_{3}, 1, x_{1}\right]=1 .
\end{aligned}
$$

S. $8 \quad P\left[0, x_{4}, x_{3}, 0, x_{1}\right] \bmod 2=2 \bmod 2=0$. $P\left[1, x_{4}, x_{3}, 0, x_{1}\right] \bmod 2=2 \bmod 2=0$, $P\left[0, x_{4}, x_{3}, 1, x_{1}\right] \bmod 2=4 \bmod 2=0$, therefore go to S.8a.
S.8a $N_{4}\left[0, x_{4}, x_{3}, 0, x_{1}\right]=1=P\left[0, x_{4}, x_{3}, 0, x_{1}\right] / 2$,
$N_{1}\left[0, x_{4}, x_{3}, 0, x_{1}\right]=1=P\left[0, x_{4}, x_{3}, 0, x_{1}\right] / 2$.
$N_{3}\left[1, x_{4}, x_{3}, 0, x_{1}\right]=1=P\left[1, x_{4}, x_{3}, 0, x_{1}\right] / 2$.
$N_{3}\left[0, x_{4}, x_{3}, 1, x_{1}\right]=2=P\left[0, x_{4}, x_{3}, 1, x_{1}\right] / 2$.
$N_{1}\left[0, x_{4}, x_{3}, 1, x_{1}\right]=2=P\left[0, x_{4}, x_{3}, 1, x_{1}\right] / 2$.
therefore go to S.8b.
S.8b $\quad \operatorname{Adj}_{4}\left[0, x_{4}, x_{3}, 0, x_{1}\right]=0 \neq N_{4}\left[0, x_{4}, x_{3}, 0, x_{1}\right]$,
$\operatorname{Adj} j_{1}\left[0, x_{4}, x_{3}, 0, x_{1}\right]=0 \neq N_{1}\left[0, x_{4}, x_{3}, 0, x_{1}\right]$,
$\operatorname{Adj} j_{3}\left[1, x_{4}, x_{3}, 0, x_{1}\right]=1=N_{3}\left[1, x_{4}, x_{3}, 0, x_{1}\right]$.
$\operatorname{Adj} j_{3}\left[0, x_{4}, x_{3}, 1, x_{1}\right]=1 \neq N_{3}\left[0, x_{4}, x_{3}, 1, x_{1}\right]$,
$\operatorname{Adj} j_{1}\left[0, x_{4}, x_{3}, 1, x_{1}\right]=1 \neq N_{1}\left[0, x_{4}, x_{3}, 1, x_{1}\right]$.
Now, $A d j_{3}\left[1, x_{4}, x_{3}, 0, x_{1}\right]=1=N_{3}\left[1, x_{1}, x_{3}, 0, x_{1}\right]$, therefore $x_{3}$ is the node variable at level 3.
S. $9 \quad \ell=4$, novars $=2$,
$\ell<5$, therefore expand the FPRM expansions about variable $x_{3}$.
$f\left(0, x_{4}, 0,0, x_{1}\right)=1 \oplus x_{4} x_{1}$
$\pi\left(0, x_{4}, 0,0, x_{1}\right) \oplus f\left(0, x_{4}, 1,0, x_{1}\right)=0$

$$
\begin{aligned}
& f\left(1, x_{4}, 0,0, x_{1}\right)=1 \\
& f\left(1, x_{4}, 0,0, x_{1}\right) \oplus f\left(1, x_{4}, 1,0, x_{1}\right)=1 \\
& f\left(0, x_{4}, 0,1, x_{1}\right)=1 \oplus x_{1} \\
& f\left(0, x_{4}, 0,1, x_{1}\right) \oplus f\left(0, x_{4}, 1,1, x_{1}\right)=1 \oplus x_{4} x_{1} \\
& f\left(1, x_{4}, 0,1, x_{1}\right)=1 \oplus x_{4} \\
& f\left(1, x_{4}, 0,1, x_{1}\right) \oplus f\left(1, x_{4}, 1,1, x_{1}\right)=x_{1}
\end{aligned}
$$

Repeating S.5 - S.8b indicates that the node variable at level 4 may be chosen arbitrarily. Hence, select $x_{4}$ as the node variable.
S. $9 \quad l=5$, novars $=1 . l=n$ therefore go to S.10.
S. $10 x_{1}$ is the node variable at level 5.

The variable ordering (from level 5 to the root of the ORMBDD) is <1,4,3,5,2>.
The ORMBDD of the FPRM expansion with this variable ordering is shown in Figure 7.8.


Figure 7.8: RORMBDD of Example 7.3

## (End or example)

7.4 Physical Implementation of RORMBDDs using Reed-Muller Universal Loglc Modules
This section details the implementation of RORMBDDs as multi-level circults
comprised of modular devices. The Reed-Muller Universal Logic Module [3. 88, 89] is the counterpart of the digital multiplexer used to implement Boolean functions expressed in sum-of-products form [120]. As demonstrated in chapter 2, RM-ULMs can be used either as individual modules or connected as networks [89] to implement logic functions represented as generalised Reed-Muller expansions. Single control input RM-ULMs (defined as RM-ULM(1)s) may be used to implement FPRM expansions represented by RORMBDDs in a manner similar to implementing BDDs by interconnecting single control input multiplexers [116].


Figure 7.9: (a) Circuit implementation of a RM-ULM(1), (b) Symbol of a RMULM(1), (c) Single node of a RMBDD

Each node of a RORMBDD may be replaced by a RM-ULM(1), where the node variable is used as the control input of the RM-ULM(1), and the two data inputs, $b_{0}$ and $b_{1}$, are connected to the input branches (Figure 7.3). The RM-ULM(1) implementation of the RORMBDD given in Figure 7.8 is displayed in Figure 7.10. Additionally, this diagram fllustrates that the number of modules in the network equals the number of nodes in the RORMBDD. However, the number of modules may be reduced by exploiting the fact that a RM-ULM(1) can implement any FPRM expansion or two variables (chapter 2). Modules which replace nodes which have both inputs connected to leaves may be deleted and the data inputs of the modules in the level above modified according to the value of the leaves. When calculating the number of modules in a RM-ULM(1) network from the RORMBDD, nodes which have both inputs connected to leaves should not be included in the node count. This is illustrated in Figure 7.11, which shows the reduced RM-ULM(1) implementation of the RORMBDD of Figure 7.8.


Figure 7.10: RM-ULM(1) implementation of the RORMBDD of Example 7.3


Figure 7.11: Reduced RM-ULM(1) Implementation of the RORMBDD of Example 7.3

An algorithm which derives good though not necessarlly optlmum RM-ULM networks was presented by Xu, Almaini, Miller and McKenzie [89]. This algorithm searches initlally for cascade networks and looks for branches
which terminate early. Unlike the RM-ULM(1) networks constructed from RORMBDDs where the control variable is assigned for each device on a particular level of the network, this algorithm does not place constraints upon the variable ordering. A disadvantage of the resulting unordered structure is that it can reduce the possibility of sharing subbranches and hence minimising the number of modules in the RM-ULM network. This problem is addressed by using RORMBDDs as a means of representing RM-ULM networks thus constraining the variable ordering. The algorithm detailed in the previous section may be employed to determine the variable ordering for the RORMBDD. The order of the variables in the lowest two levels of the RORMBDD, level ( $n-1$ ) and level ( $n$ ), does not affect the number of modules in the RM-ULM network hence it is only necessary to perform ( $n-2$ ) iterations of the algorithm. Additionally, $n!/ 2$ variable orderings should be evaluated when exhaustively searching for the optimum RM-ULM network.
7.5 The Reed-Muller Factored Form and Gate Level Implementation of RORMBDDs
The RORMBDD may also be used to determine RM factored forms and hence multi-level circuits composed of AND gates, EXOR gates and Inverters.

The RM factored form of any ESOP expression was first described by Saul [85]. The structure of the RORMBDD means that a factored form of the FPRM expansion which is represented may be easily derived by tracing path from each terminal with value 1 to the root of the RORMBDD. This is illustrated in the following example.

Example 7.4 Derive the factored form of the FPRM expansion

$$
\begin{aligned}
f_{0}\left(x_{3}, x_{4}, x_{3}, x_{2}, x_{1}\right)= & 1 \odot x_{2} \oplus x_{2} x_{1} \oplus x_{3} x_{2} \oplus x_{4} x_{1} \oplus x_{4} x_{3} x_{2} x_{1} \oplus x_{3} \odot x_{3} x_{2} \\
& \bullet x_{3} x_{3} \oplus x_{3} x_{3} x_{2} x_{1} \oplus x_{3} x_{4} x_{2}
\end{aligned}
$$

The RORMBDD of this expression is illustrated in Figure 7.8 and the factored form may be derived from this structure.

The RM factored form is

$$
f_{0}\left(x_{3}, x_{4}, x_{3}, x_{2}, x_{1}\right)=1 \oplus x_{4} x_{1} \oplus x_{5}\left(1 \odot x_{3}\right) \oplus x_{2}\left(1 \odot x_{1} \bullet x_{3}\left(1 \oplus x_{4} x_{1}\right) \oplus x_{5}\left(1 \odot x_{4} \odot x_{3} x_{1}\right)\right)
$$

(End of example)

The implementation of RORMBDDs using RM-ULM(1)s has been described in the preceding section. It is also possible to directly derive a circuit comprised of AND gates, EXOR gates and inverters from a RORMBDD. This process is now described.

The inputs to the circuit are the decision variables of the nodes of the RORMBDD. Non-terminal nodes with both input branches connected to terminal nodes form input variables to the circuit.

A non-terminal node with both input branches connected to terminal nodes with the value 1 represents an input to the circuit. The input variable is the decision variable of the node and should be inverted. A non-terminal node with the 0 -input branch connected to a terminal node with the value 0 and the 1 -input branch connected to a terminal node with the value 1 also represents an input to the circuit. The input variable is the decision variable of the node.

A non-terminal node a with the 0 -input branch connected to a non-terminal node $b$ and the 1 -input branch connected to a terminal node with the value 1 may be replaced by an EXOR gate. The Inputs to the gate are the decision variable of the node a and the output of the subcircult rooted at node $b$. A non-terminal node a with the 0 -input branch connected to a terminal node with the value 0 and the l-input branch connected to a nonterminal node $b$ may be replaced by an AND gate. The Inputs to the gate are the decision variable of the node $a$ and the output of the subcircult rooted at node $b$.

A non-terminal node a with the 0 -input branch connected to a terminal node with the value 1 and the 1 -input branch connected to a non-terminal node $b$ may be replaced by an AND gate and an EXOR gate. The inputs to the AND gate are the decision variable of the node $a$ and the output of
the subcircuit rooted at node $b$. The inputs to the EXOR gate are the output of the AND gate and the Boolean constant 1. Alternatively, the EXOR gate may be replaced by an inverter.

Finally, a non-terminal node a with the 0 -input branch connected to a nonterminal node $b$ and the 1 -input branch connected to a non-terminal node $c$ may be replaced by an AND gate and an EXOR gate. The inputs to the AND gate are the outputs of the subcircuits rooted at nodes $b$ and $c$. The inputs to the EXOR gate are the output of the AND gate and decision variable of node a.

Thus, any RORMBDD may be represented using discrete gates as opposed to the modular representation described previously.

### 7.6 Summary

RORMBDDs have been introduced as a graphical means of representing FPRM expansions. Analogies have been drawn between these structures and ROBDDs used to represent Boolean functions. An algorithm has been presented which may be employed to derive optimal (sub-optimal) RORMBDDs representing FPRM expansions. The algorithm determines the order of the FPRM expansion variables from the root to terminal nodes of the RORMBDD. The use of this algorithm to form RORMBDDs from which efficient RM-ULM(1) networks can be constructed has been described. Additionally, the derivation of the RM factored form from a RORMBDD representation of a FPRM expansion has been described and the gate level implementation of these structures discussed.

## Chapter 8

## Conclusions

The aims of this research project were to develop techniques suitable for representing, generating and minimising different classes of ESOP forms. These techniques have been fully described in this thesis and are now briefly summarised. The outcome of this research project is also reviewed and areas suitable for further research are suggested.

### 8.1 Review of Algorithms and Techniques

Heuristic minimisation algorithms suitable for deriving optimal (sub-optimal) FPRM expansions from initial Boolean functions have been developed. These algorithms were derived from an existing method through the introduction of a variety of modifications. The aim of thls work was to improve the effectiveness of the original technique. The benefits of each of these modifications were explored and illustrated with results obtained through minimising sets of Boolean functions. Thus, a group of improved heuristic minimisation algorithms have evolved which may be employed to optimise FPRM expansions.

An incompletely specified Boolean function may be minimised by assigning appropriate values to the 'don't care' terms of the function. An algorlthm has been developed which may be employed to transform an incompletely specified Boolean function to a FPRM expansion comprised of specified and 'don't care' terms. The 'don't care' terms may then be assigned values so as to maximally reduce the number of product terms in the FPRM expansion, where the polarity of the expansion is pre-determined. The technique developed for this purpose is non-exhaustive, nowever, the number of evaluations which must be made before the optimum solution is derived is dependent on the inherent structure of the initial incompletely specified Boolean function. Although these technlques have been proposed as an extension to the tabular technique their use is not limited to this method. Indeed, these techniques are versatile and may be used In conjunction with other methods for representing and generating FPRM expansions from Boolean functions e.g. Habib [41], Harking [42]. Additionally, two heuristic
algorithms have been developed which may be used to determine optimum (sub-optimum) FPRM expansions of incompletely specified Boolean functions. The effectiveness of these algorithms was evaluated and their performance illustrated through minimising sample sets of Boolean functions.

Kronecker expansions are a broader class of ESOP forms and include FPRM expansions as a subset. A technique has been developed which may be employed to generate KRO expansions from incompletely specified Boolean functions. Additionally, the problem of generating ESOP forms from reduced Boolean functions has been addressed. This led to the development of an algorithm which derives KRO expansions from reduced Boolean SOP forms where the product terms of the initial Boolean expression are disjoint. The value of this technique lies in its ability to generate KRO expansions without having to first derive the canonical form of the Boolean expression. Both these methods employ modified forms of the original tabular representation and are valuable extensions to the scope of the tabular technique.

The algorithms and techniques which have already been summarised may be employed to represent and optimise FPRM and KRO expansions. These expressions are two-level ESOP forms and may be implemented as two-level networks of discrete gates or RM-ULMs, or Indeed, using programmable logic devices comprised of an AND array and an EXOR array (XPLAS). However, a switching function may also be efficiently reallsed as a multilevel circuit. RORMBDDs have been developed as an alternative graphical means of representing and deriving multi-level realisations of FPRM expansions. The RORMBDD representing a particular FPRM expansion is a canonical representation and the number of nodes in that RORMBDD may only be reduced by altering the order of the variables in the structure. The synthesis of RORMBDDS may be regarded as comprising of two distinct steps, the first step is the derivation of minimal FPRM expansions whilst the second step is the determination of 'good' varlable orderings. A heuristic algorithm has been developed which derives a minimal (subminimal) RORMBDD through variable ordering. RORMBDDs allow subfunction sharing and may also be 'flattened' to derive two-level representations. The use of RORMBDDs has also been extended to include the representation of ESOP forms. The construction of RM-ULM(1) networks from RORMBDDs was
described and demonstrated a practical application of this work.

An arbitrary Boolean function may be represented by $2^{n}$ FPRM expansions. $3^{n}$ KRO expansions or, at most, $3^{\text {tn }}$ ESOP forms, where $n$ is the number of variables and $t$ the number of product terms. It is, therefore, most probable that the optimum ESOP form representing a Boolean function will belong to the class of ESOP forms as opposed to the subclasses of FPRM and KRO expansions. This observation does not devalue the techniques and heuristic algorithms developed in this thesis. FPRM and KRO expansions are small, well-defined classes of ESOP forms and the optimum (sub-optimum) FPRM expansion of any Boolean function may be consistently identifled within a practical time-scale. It is then possible to utilise this expression as a starting point for further minimisation resulting in ESOP forms.

The algorithms and techniques developed during this research project may be employed to represent, generate and minimise FPRM and KRO expansions from incompletely specified Boolean functions. Although the performance of each of these techniques is by no means outstanding, they form a useful set of reliable synthesis tools.

### 8.2 Further Research

It is possible to identify areas in which the algorithms and techniques described in this thesis could be improved. These are now highlighted as areas for further research. Additionally, more general aspects of logic synthesis using ESOP forms are considered.

- It may be possible to improve the efficiency of the technique which assigns the 'don't care' terms of an incompletely specifled Boolean function to derive an optimum FPRM expansion of pre-determined polarity. This could be implemented by determining the most efficient order in which to evaluate the RM 'don't care' terms.
- The tabular technique for generating KRO expansions from incompletely specified Boolean functions could be readily extended to allow the realisation of PSDKRO expansions. This is a simple task requiring only that the tables representing KRO expansions be divided into sub-tables as varlables are transformed into the
required polarity. This would further increase the value of the tabular technique.
- A heuristic minimisation method which derives optimal (sub-optimal) KRO expansions of incompletely specified Boolean functions would be most valuable. The algorithm should exhibit improved performance when evaluated against existing techniques. This work could also be extended to include ESOP forms.
- The problem of determining the variable ordering in RORMBDDs so as to construct efficient multi-level representations requires further attention.

The implementation of RORMBDDs using RM-ULMs with multiple control inputs (e.g. RM-ULM(2)) should be investigated.

- Although synthesis techniques based on the algebra of $\mathbf{G F}(2)$ are effective, not all switching functions may be efficiently represented as ESOP forms. A more practical approach would seem to be that of the 'mixed' synthesis system where functions are represented using the OR and EXOR operators and optimised using both Boolean and RM techniques. This is an area of particular interest where there is much scope for the development of novel techniques.


## List of References

1. ALMAINI, A.E.A., THOMSOM, P., HANSON, D. : 'Tabular Techniques for Reed-Muller Logic', Int. J. Electronics, 70, (1), 1991, pp. 23-34
2. BLAHUT, R.E. : 'Theory and Practice of Error Control Codes', Addison-Wesley Publishing Company, 1983
3. GREEN, D. : 'Modern Logic Design', Addison-Wesley Publishing Company, 1986
4. ALMAINI, A.E.A. : 'Electronic Logic Systems', 2nd Edition, Prentice Hall International, 1989
5. SASAO, T., BESSLICH, P. : 'On the Complexity of Mod-2 Sum PLA's', I.E.E.E. Trans. Comp., C-39, (2), 1990, pp. 262-266
6. SASAO, T. : 'EXMIN: A Simplification Algorithm for Exclusive-OR-Sum-Of-Products Expressions for Multiple-Valued Input Two-Valued Output Functions', Proc. 20th Int. Symp. on Multiple-Valued Logic, 1990, pp.128-135
7. SASAO, T. : 'Logic Synthesis and Optimisation', Kluwer Academic Publishers, 1993
8. SAUL, J. : 'Logic Synthesis for Arithmetic Circuits Using the ReedMuller Representation', Proc. European Conf. on Design Automation, (EDAC'92), 1992, pp. 109-113
9. REDDY. S.M. : 'Easily Testable Realizations for Logic Functions', I.E.E.E. Trans. Comp., C-21, (11), 1972, pp. 1183-1188
10. DAMARLA, T., KARPOVSKY, M. : 'Detection of Stuck-at and Bridging Faults in Reed-Muller Canonical (RMC) Networks', I.E.E. I'roc. E., 136. (5), 1989, pp. 430-433
11. LOUREIRO, G.V. : 'Digital Systems Design for Testabllity based on a Reed-Muller Tree-Circuit Approach', Ph.D. Thesis, Univ. Manchester Institute of Science and Technology, U.K., 1993
12. GREEN, D.H. : 'Reed-Muller Expansions with Fixed and Mixed Polarlties over GF(4)', I.E.E. Proc. E., 137, (5), 1990, pp. 380-388
13. GREEN, D.H., EDKINS, M. : 'Synthesis Procedures for Switching Circuits Represented in Generalised Reed-Muller Form over a Finlte Field', I.E.E. Proc. E., 1 (1), 1978, pp. 27-35
14. WHITAKER, S. : 'Pass-transistor Networks Optimise n-MOS Loglc', Electronics, September 22, 1983, pp. 144-148
15. SVENSSON, C., TJARNSTROM, R. : 'Switch-level Simulation and the Pass Transistor Exor Gate', I.E.E.E. Trans. Computer-Aided Design. CAD-7, (9), 1988, pp. 994-997
16. FROESSL, J., ESCHERMANN, B. : 'Module Generation for AND/XORFields (XPLAs)', Proc. I.E.E.E. Int. Conf. on Computer Design : VLSI in Computers and Processors, 1991, pp. 26-29
17. CSANKY, L., PERKOWSKI, M.A., SCHAFER, I. : 'Canonical Restricted Mixed-polarity Exclusive-OR Sums of Products and the Efficient Algorithm for their Minimisation', I.E.E. Proc. E., 140, (1), 1993, pp. 69-77
18. PEARCE, M., SAUL, J., LESTER, N. : 'Multi-level Logic Synthesis for Concurrent Logic CLi6000 Devices', Proc. I.E.E. Colloquium on Field Programmable Gate Arrays - Technology and Applications, 1993, pp. 10/1-10/4
19. ACTEL : 'The FPGA Design Guide', Actel, 1991
20. SAUL, J.M. : 'Towards a Mixed Exclusive-/Inclusive-OR Factored Form', Proc. I.F.I.P. W.G. 10.5 Workshop on Applications of the ReedMuller Expansion in Circuit Design', 1993, pp. 2-5
21. MARINKOVIC, S.B., TOSIC, Z. : 'Algorithm for Minimal Polarised Polynomial Form Determination', I.E.E.E. Trans. Comp., C-23, (12), 1974, pp. 1313-1315
22. DAVIO, M., DESCHAMPS, J.-P., THAYSE, A. : 'Discrete and Switching Functions', McGraw-Hill International Book Company, 1978
23. GRIMALDI, R.P. : 'Discrete and Combinatorial Mathematics', AddisonWesley Publishing Company, 1985
24. MULLER, D.E. : 'Application of Boolean Algebra to Switching Circuit Design and to Error Detection', I.R.E. Trans. Electronic Computers, EC-3, 1954, pp. 6-12
25. REED, I.S. : ‘A Class of Multiple-Error-Correcting Codes and the Decoding Scheme', I.R.E. Trans. Information Theory, IT-4, 1954, [p.3849
26. SHANNON, C.E. : 'A Symbolic Analysis of Relay and Switching Circuits', Trans. American Institute of Electrical Engineers, 57, 1938, pp. 713-723
27. GREEN, D.H. : 'Familles of Reed-Muller Canonical Forms', Int. J. Electronics, 70, (2), 1991, pp. 259-280
28. COHN, M. : 'Inconsistent Canonical Forms of Switching Functions'. I.R.E. Trans. Electronic Computers, EC-11, 1962, pp. 284-285
29. LEE, C.Y. : 'Representation of Switching Circults by Binary-Decision Programs', Bell Syst. Tech. J., 38, (4), 1959, pp. 385-999
30. AKERS, S.B. : 'Binary Decision Dlagrams', I.E.E.E. Trans. Comp., C-27, (6), 1978, pp. 509-516
31. BRYANT, R.E. : 'Graph-Based Algorithms for Boolean Function Manipulation', I.E.E.E. Trans. Comp., C-35, (8), 1986, pp. 677-691
32. FRIEDMAN, S.J., SUPOWIT, K.J. : 'Finding the Optimal Variable Ordering for Binary Decision Diagrams', I.E.E.E. Trans. Comp., 39, (5), 1990, pp. 710-713
33. FABRICIUS, E.D. : 'Introduction to VLSI Design', McGraw-Hill Publishing Company, 1990
34. MCKENZIE, L., ALMAINI, A.E.A, MILLER, J.F., THOMSON, P. : 'Optimisation of Reed-Muller Logic Functions', Int. J. Electronics, 75, (3). 1993, pp. 451-466
35. WU, X., CHEN, X., HURST, S.L. : 'Mapping of Reed-Muller Coefficients and the Minimisation of Exclusive-OR Switching Functions', I.E.E. Proc. E, 129, (1), 1982, pp. 15-20
36. TRAN, A. : 'Graphical Method for the Conversion of Minterms to Reed-Muller Coefficients and the Minimisation of Exclusive-OR Switching Functions', I.E.E. Proc. E, 134, (2), 1987, pp. 93-99
37. GREEN, D.H. : 'Reed-Muller Expansions of Incompletely Specified Functions', I.E.E. Proc. E., 134, (5), 1987, pp. 228-236
38. TRAN, A. : 'Tri-state Map for the Minimisation of Exclusive-OR Switching Functions', I.E.E. Proc. E, 136, (1), 1989, pp. 16-21
39. BESSLICH, Ph.W. : 'Efficient Computer Method for ExOR Logic Design', I.E.E. Proc. E, 130, (6), 1983, pp. 203-206
40. HABIB, M.K. : 'New Approach for the Generation of Minimal ReedMuller Exclusive-OR Expansions with Mixed Polarity', Int. J. Electronics, 66, (6), 1989, pp. 865-874
41. HABIB, M.K. : 'Boolean Matrix Representation for the Conversion of Minterms to Reed-Muller Coefficients and the Minimisation of Exclusive-0R Switching Functions', Int. J. Electronics, 68, (4), 1930. pp. 493-506
42. HARKING, B. : 'Efficient Algorithm for Canonlcal Reed-Muller Expansions of Boolean Functions', I.E.E. Proc. E., 137. (5), 1930, pp. 366-370
43. GREEN, D.H., TAYLOR, I.S. : 'Modular Representation of Multiple-valued Logic Systems'. I.E.E. Proc. E, 121, (6), 1974, pp. 409-418
44. ZHANG, Y.Z., RAYNER, M.A. : 'Minimisation of Reed-Muller Polynomials with Fixed Polarity', I.E.E. Proc. E, 131, (5), 1984, pp. 177-186
45. SALUJA, K.K., ONG, E.H. : 'Minimisation of Reed-Muller Canonic Expansion', I.E.E.E. Trans. Comp., C-28, (7), 1979, pp. 535-537
46. SARABI, A., PERKOWSKI, M.A. : 'Fast Exact and Quasi-minimal Minimisation of Highly Testable Fixed-Polarity AND/XOR Canonical Networks', Proc. 29th Design Automation Conf., 1992, pp. 30-35
47. PURWAR, S. : 'An Efficient Method of Computing Generalised Reed-Muller Expansions from Binary Decision Diagram', I.E.E.E. Trans. Comp., 40, (11), 1991, pp. 1298-1301
48. KEBSCHULL, U., SCHUBERT, E., ROSENSTIEL, W. : 'Multilevel Logic Synthesis Based on Functional Decision Diagrams', Proc. European Conf. on Design Automation, (EDAC'92), 1992, pp. 43-47
49. SCHUBERT, E., KEBSCHULL, U., ROSENSTIEL, W. : 'FDD Based Technology Mapping for FPGA', Proc. EuroASIC'92, 1992, pp. 14-18
50. MILLER, J.F., THOMSON, P. : 'Highly Efficient Exhaustive Search Algorithms for Optimising Canonical Reed-Muller Expansions of Boolean Functions', Int. J. Electronics, 76, (1), 1994, pp. 37-56
51. MUKHOPADHYAY, A., SCHMITZ, G. : 'Minimisation of Exclusive-OR and Logical Equivalence Switching Circuits', I.E.E.E. Trans. Comp., C-19, (2), 1970, pp. 132-140
52. LUI, P.K., MUZIO, J.C. : 'Boolean Matrix Transforms for the Minimisation of Modulo-2 Canonical Expansions', I.E.E.E. Trans. Comp., 41, (3), 1992, pp. 342-347
53. UNGERN, E. : 'Minimisation of Exclusive-OR Switching Clrcults', I.E.E.E. Repository, R-72-79, 1972, 25 pages
54. CLARKSON, T.G., ZHUANG, N. : 'A Vector Algorlthm for Reed-Muller Expansions', I.E.E. Electronics Letters, 30, (7), 1994, pp. 543-550
55. CSANKY, L., PERKOWSKI, M., SCHAEFER, I. : 'Canonical Restricted Mixed-Polarity Exclusive Sums of Products'. Proc. lnt. Symp. on Circuits and Systems (I.S.C.A.S.'92), 1992, pp. 17-20
56. LUI, P.K., MUZIO, J.C. : 'Simplified Theory of Boolean Functions', Int. J. Electronics, 68, (3), 1990, pp. 329-341
57. LUI, P.K., MUZIO, J.C. : 'Boolean Matrix Transforms for the Iarlty Spectrum and Minimisation of Modulo-2 Canonical Expansions', I.E.E: Proc. E, 138, (6), 1991, pp. 411-417
58. LUI, P.K., MUZIO, J.C. : 'Structure of Modulo-2 Ring-sum Canonical Expansion for Boolean Functions', Int. J. Electronics, 72, (1), 1932, pp. 21-35
59. FALKOWSKI, B.J., PERKOWSKI, M.A. : 'On the Calculation of Cenerallsed Reed-Muller Canonical Expansions from Disjoint Cube Representations of Boolean Functions', Proc. 33rd Midwest Symp. on Circults and Systems, 1990, pp. 1131-1134
60. FALKOWSKI, B.J., PERKOWSKI, M.A. : 'One More Way to Calculate Generalised Reed-Muller Expansions of Boolean Functions', Int. J. Electronics, 71, (3), 1991, pp. 385-396
61. RIEGE, M.W., BESSLICH, Ph.W. : 'Low-complexity Synthesis of Incompletely Specified Multiple-output Mod-2 Sums', I.E.E. Proc. E, 139, (4), 1992, pp. 355-362
62. VARMA, D., TRACHTENBERG, E.A. : 'Computation of Reed-Muller Expansions of Incompletely Specified Boolean Functions from Reduced Representations', I.E.E. Proc. E, 138, (2), 1991, pp. 85-92
63. BRAYTON, R.K., HACHTEL, G.D., MCMULLEN, C.T. SANGIOVANNIVINCENTELLI, A.L. : 'Logic Minimization Algorithms for VLSI Synthesis', Kluwer Academic Publishers, 1985
64. BIOUL, G., DAVIO, M., DESCHAMPS, J.P. : 'Minimisation of Ring-Sum Expansions of Boolean Functions', Philips Rescarch Reports, 28, 1973, pp. 17-36
65. GREEN, D.H. : 'Reed-Muller Canonical Forms with Mixed Polarity and their Manipulations', I.E.E. Proc. E, 137, (1), 1990, pp. 103-113
66. GREEN, D.H., KHUWAJA, G.A. : 'Simplification of Switching Functions Expressed in Reed-Muller Algebralc Form', I.E.E. Proc. E. 139, (6), 1992, pp. 511-518
67. FLEISHER, H., TAVEL, M., YEAGER, J. : 'Exclusive-OR Representations of Boolean Functions', I.B.M. Journal of Research and Development, 27. (4), 1983, pp. 412-415
68. SARABI, A., HO, P.F., IRAVANI, K., DAASCH, W.R., PERKOWSKI, M.A. : 'Minimal Multi-level Realisation of Switching Functions based on Kronecker Functional Decision Dlagrams'. Proc. Int. Workshop on Logic Synthesis, 1993, pp. 3al-3a6
69. ROBINSON, J.P., YEH, C.-L. : 'A Method for Modulo-2 Minimisation'. I.E.E.E. Trans. Comp., C-31, (8), 1982, pp. 800-801
70. EVEN, S., KOHAVI, I., PAZ, A. : 'On Minimal Modulo 2 Sums of Products for Switching Functions', I.E.E.E. Trans. Comp., EC-14, 1967, pp. 671674
71. HELLIWELL, M., PERKOWSKI, M. : A Fast Algorithm to Minimisc Multi-Output Mixed Polarity Generalised Reed-Muller Forms', Proc. 25th A.C.M./I.E.E.E. Design Automation Conference, 1988, pp. 427-432.
72. SAUL, J.M. : 'An Improved Algorithm for the Minimisation of Mixed Polarity Reed-Muller Representations', Proc. Int. Conf. on Computer-Aided Design : VLSI in Computers and lrocessors (I.C.C.D.'90), 1990, pp. 372-375
73. PITTY, E.B., SALMON, J.V. : 'Input Irredundancy of Mlxed-polarity Reed-Muller Equations', Electronics Letters, 24, (5), 1988, pp. 258-260
74. FLEISHER, H., TAVEL, M., YEAGER, J. : 'A Computer Algorithm for Minimising Reed-Muller Canonical Forms', I.E.E.E. Trans. Comp., C-36, (2), 1987, pp. 247-250
75. PAPAKONSTANTINOU, G. : 'Minimisation of Modulo-2 Sum of Products', I.E.E.E. Trans. Comp., C-28, (2), 1979, pp. 163-167
76. SALMON, J.V., PITTY, E.B., ABRAHAMS, M.S. : 'Syntactic Translation and Logic Synthesis in Gatemap, I.E.E. Proc. E, 136, (4), 1989, pp. 321-328
77. DE GEUS, A.J., COHEN, W. : 'A Rule-Based System for Optimizing Combinational Logic', I.E.E.E. Design and Test of Computers, 1985, pp. 22-32
78. TRAN, A., WANG, J. : 'Decomposition Method for Minimisation of Reed-Muller Polynomials in Mixed Polarity', I.E.E. Proc. E, 140, (1), 1993, pp. 65-68
79. LIN, T.-L., TRAN, A. : 'Minimisation of Multiple-output Exclusive-OR Switching Functions', Int. J. Electronics, 75, (4), 1993, pp. 665-674
80. GREEN, D.H., KHUWAJA, G.A. : ‘Tabular Simplification Method for Switching Functions Expressed in Reed-Muller Algebraic Form', Int. J. Electronics, 75, (2), 1993, pp. 297-314
81. TRAN, A., LEE, E. : 'Generalisation of Tri-state Map and a Composition Method for Minimisation of Reed-Muller Polynomials in Mixed Polarity', I.E.E. Proc. E, 140, (1), 1993, pp. 59-64
82. PERKOWSKI, M., CHRZANOWSKA-JESKE, M. : ‘An Exact Algorithm to Minimise Mixed-Radix Exclusive Sums of Products for Incompletely Specified Boolean Functions', Proc. Int. Symp. on Circuits and Systems (I.S.C.A.S.'90), 1990, pp. 1652-1655
83. FISHER, L.T. : 'Unateness Properties of AND-EXCLUSIVE-OR Logic Circuits', I.E.E.E. Trans. Comp., C-23, (2), 1974, pp. 166-172
84. FALKOWSKI, B.J. and PERKOWSKI, M.A. : ‘Algorithm for the Generation of Disjoint Cubes for Completely and Incompletely Specified Boolean Functions', Int. J. Electronics, 70, (3), 1991, pp. 533-538
85. SAUL, J. : 'Logic Synthesis Based on the Reed-Muller Representation', Ph.D. Thesis, Univ. Bristol, U.K., 1992
86. MCKENZIE, L., XU, L., ALMAINI, A. : 'Graphical Representation of Generalised Reed-Muller Expansions', Proc. IFIP WG 10.5 Workshop On Applications of the Reed-Muller Expansion in Circuit Design, 1993, pp. 181-187
87. XU, L., MCKENZIE, L. : 'Multi-level Optimisation of Fixed Polarity Reed-Muller Expansions using Reed-Muller Binary Decision Diagrams', Proc. I.E.E. Colloquium on Synthesis and Optimisation of Logic Systems, 1994, pp. 3/1-3/4
88. XU L., ALMAINI, A.E.A. : 'Full-custom Design of Reed-Muller Universal Logic Modules', Int. J. Electronics, 74, (4), 1993, pp. 605-613
89. XU, L., ALMAINI, A.E.A., MILLER, J.F., MCKENZIE, L. : 'Reed-Muller Universal Logic Module Networks', I.E.E. Proc. E, 140, (2), 1993, pp. 105-108
90. EDWARDS, M.D. : 'Automatic Logic Synthesis Techniques for Digital Systems'. MacMillan New Electronics, 1992
91. BRAYTON, R.K., HACHTEL, G.D., SANGIOVANNI-VINCENTELLI, A.L. : 'Multilevel Logic Synthesis', Proc. I.E.E.E., 78, (2), 1990, pp. 264-300
92. BRAYTON, R.K., MCMULLEN, C. : 'Synthesis and Optimisation of Multistage Logic', Proc. Int. Conf. Computer Design (I.C.C.D.'84), 1984, pp. 23-28
93. BRAYTON, R.K., MCMULLEN, C. : 'The Decomposition and Factorisation of Boolean Expressions', Proc. Int. Symp. Circuits and Systems (I.S.C.A.S.'82), 1982, pp. 49-54
94. BRAYTON, R.K. : 'Factoring Logic Functions', I.B.M. Journal of Research and Development, 31, (2), 1987, pp. 187-198
95. BRAYTON, R., RUDELL, R., SANGIOVANNI-VINCENTELLI, A., WANG, A. : 'Multi-level Logic Optimisation and the Rectangular Covering Problem', Proc. Int. Conf. Computer-Aided Design (I.C.C.A.D.'87), 1987, pp. 66-69
96. WANG, A.R.R. : 'Algorithms for Multi-level Logic Optimisation', Ph.D. Thesis, Univ. California, Berkeley, U.S.A., 1989
97. BRAYTON, R.K., RUDELL, R., SANGIOVANNI-VINCENTELLI, A., WANG, A.R. : 'MIS : A Multiple-Level Logic Optimisation System', I.E.E.E. Trans. Computer-Aided Design, CAD-6, (6), 1987, pp. 1062-1081
98. BARTLETT, K., COHEN, W., DE GEUS, A., HACHTEL, G. : 'Synthesis and Optimisation of Multilevel Logic under Timing Constraints', I.E.E.E. Trans. Computer-Aided Design, CAD-5, (4), 1986, pp. 582-596
99. BOSTICK, D., HACHTEL, G.D., JACOBY, R., LIGHTNER, M.R., MOCEYUNAS, P., MORRISON, C.R., RAVENSCROFT, D. : 'The Boulder Optimal Logic Design System', Proc. Int. Conf. Computer-Aided Design (I.C.C.A.D.'87), 1987, pp. 62-65
100. BERGAMASCHI, R.A. : 'SKOL : A System for Logic Synthesis and Technology Mapping', I.E.E.E. Trans. Computer-Aided Design, 10, (11), 1991, pp. 1342-1355
101. YKMAN-COUVREUR, C. : 'Multi-level Boolean Optimisation for Incompletely Specified Boolean Functions in PHIFACT', Proc. Int. Workshop : Logic and Architecture Synthesis for Silicon Compilers, 1988, pp. 127-152
102. KARPOVSKY, M. : 'Multi-level Logical Networks', I.E.E.E. Trans. Comp., C-36, (2), 1987, pp. 215-226
103. LAWLER, E.L. : 'An Approach to Multilevel Boolean Minimisation', J. Assoc. Comp. Mach., 11, (3), 1964 pp. 283-295
104. BRAYTON, R.K. : 'Don't Cares in Multi-level Logic Synthesis', Proc. 6th M.I.T. Conf., Advanced Research in VLSI, 1990, pp. 217-220
105. BRAYTON, R.K., SENTOVICH, E.M., SOMENZI, F. : 'Don't Cares and Global Flow Analysis of Boolean Networks', Proc. Int. Conf. Computer-Aided Design (I.C.C.A.D.'88), 1988, pp. 98-101
106. BARTLETT, K.A., BRAYTON, R.K., HACHTEL, G.D., JACOBY, R.M., MORRISON, C.R., RUDELL, R.L., SANGIOVANNI-VINCENTELLI, A., WANG, A.R. : 'Multilevel Logic Minimisation Using Implicit Don't Cares', I.E.E.E. Trans. Computer-Aided Design, 7, (6), 1988, pp. 723-740
107. GREEN, D.H., FOULK, P.W. : 'Adaptive-logic Trees for use in Multilevel-circuit Design', Electronics Letters, 5, (5), 1969, pp. 83-84
108. SAUL, J.M. : 'An Algorithm for the Multi-level Minimisation of Reed-Muller Representations', Proc. Int. Conf. Computer Design, (I.C.C.D.'91), 1991, 4 pages
109. PERKOWSKI, M., DRISCOLL, M., LUI, J., SMITH, D., BROWN, J., YANG, L., SHAMSAPOUR, A., HELLIWELL, M., FALKOWSKI, B., WU, P., CIESIELSKI, M., SARABI, A. : 'Integration of Logic Synthesis and High-Level Synthesis into the Diades Design Automation System', Proc. 22nd I.E.E.E. Int. Symp. Circuits and Systems, (I.S.C.A.S.'89), 1989, pp. 748-751
110. MATSUNAGA, Y., FUJITA, M. : 'Multilevel Logic Optimisation Using Binary Decision Diagrams', Proc. I.E.E.E. Int. Conf. Computer-Aided Design, (I.C.C.A.D.'89), 1989, pp. 556-559
111. FUJITA, M., MATSUNAGA, Y., KAKUDA, T. : 'On Variable Ordering of Binary Decision Diagrams for the Application of Multi-level Logic Synthesis', Proc. European Conf. Design Automation (EDAC'91), 1991, pp. 50-54
112. ISHIURA, N., SAWADA, H., YAJIMA, S. : 'Minimisation of Binary Decision Diagrams Based on Exchanges of Variables', Proc. I.E.E.E. Int. Conf. Computer-Aided Design (I.C.C.A.D.'91), 1991, pp. 472-475
113. BESSON, T., BOUZOUZOU, H., FLORICICA, I., SAUCIER, G., ROANE, R. : 'Input Order for ROBDDs Based on Kernel Analysis, Proc. European Conf. Design Automation (E.D.A.C./EuroASIC'93), 1993, 7 pages
114. BESSON, T., BOUZOUZOU, H., CRASTES, M., FLORICICA, I., SAUCIER, G. : 'Synthesis on Multiplexer-based FPGA using Binary Decision Diagrams', Proc. Int. Conf. Computer Design (I.C.C.D.'92), 1992, 5 pages
115. LIAW, H.-T., LIN, C.-S. : 'On the OBDD-Representation of General Boolean Functions', I.E.E.E. Trans. Comp., 41, (6), 1992, pp. 661-664
116. MATOS, J.S., OLDFIELD, J.V. : 'Binary Decision Diagrams : From Abstract Representations to Physical Implementations', Proc. 20th Design Automation Conf., 1983, pp. 567-570.
117. ABADIR, M.S., REGHBATI, H.K. : 'Functional Test Generation for Digital Circuits Described Using Binary Decision Diagrams', I.E.E.E. Trans. Comp., C-35, (4), 1986, pp. 375-379
118. COUDERT, O., MADRE, J.C. : 'Implicit and Incremental Computation of Primes and Essential Primes of Boolean Functions', Proc. 29th Design Automation Conf. (D.A.C.'92), 1992, pp. 36-39.
119. SASAO, T. : 'Ternary Decision Diagrams and their Application', Proc. Int. Workshop on Logic Synthesis, 1993, pp. 6c1-6c11
120. YAU S.S., TANG, C.K. : 'Universal Logic Modules and their Applications', I.E.E.E. Trans. Comp., C-19, (2), 1970, pp. 141-149

# PUBLISHED PAPERS NOT INCLUDED 


[^0]:    1 The work presented in this chapter is published in [34] (see inside back cover).

[^1]:    1 The work presented in this chapter is published in [34] (see inside back cover).

[^2]:    The work presented in this chapter is published in [86, 87] (see inside back cover). L. MCKenzie developed and implemented the variable ordering algorithm whilst the work undertaken by L. Xu was mainly concerned with RM-ULM implementation.

