Tinghuai Chen

Fault Diagnosis and Fault Tolerance

A Systematic Approach to Special Topics

Springer-Verlag

Tinghuai Chen

Fault Diagnosis and Fault Tolerance

A Systematic Approach to Special Topics

With 120 Figures and 23 Tables

Springer-Verlag

Berlin Heidelberg New York
London Paris Tokyo
Hong Kong Barcelona
Budapest

Prof. Tinghuai Chen

Computer Research Institute
Chongqing University
Chongqing, PR China 630044

ISBN 3-540-54962-5 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-54962-5 Springer-Verlag New York Berlin Heidelberg

Printed in Germany

Typesetting: camera ready by author

45/3140-543210 - Printed on acid-free paper

Preface

With the rapid growth of integration scale of VLSI chips and the present need for reliable computers in space exploration, fault diagnosis and fault tolerance have become more important than before, and hence reveal a lot of interesting topics which attract many researchers to make a great number of contributions to this field. In recent years, many new and significant results have been achieved. A quick scan over the proceedings of the conferences on fault-tolerant computing and design automation as well as on testing will convince the reader of that. But unfortunately these achievements have not been entirely reflected in the textbooks, so that there seems to be a gap for the new researcher who already has the basic knowledge and wants to begin research in this area. As a remedy for this deficiency, this book is intended for beginners, especially graduate students, as a textbook which will lead them to the frontier of some branches of the fault-tolerant computing field.

The first chapter introduces the four-valued logic B_4 and its applications. In 1966 Roth first proposed this four-valued logic as a technique to generate tests for logical circuits, but this work did not concern the mathematical basis of B_4 itself. Here in this book it is considered in an abstract way, related to the representations of boolean functions in B_4, the computation of the four components of the functions, called the STAR algorithm, which plays the same role as boolean differentials and boolean differences, and also solving equations in B_4. The applications depend upon the explanations of these four elements: The first explanation can be used to derive the test set for combinational and sequential circuits, which has some advantages over existing techniques. The second explanation leads to hazard identification and then to dynamic test generation, which can detect most of the statically undetectable faults. The third explanation is the transition logic concerning the propositional and predicate calculus, as well as logical inferences, which can more or less reflect the dynamic logical behavior of the objective world.

The fault-diagnosis problem at the system level was formulated by Preparata, Metze and Chien in 1967. Since then a lot of work has been done for different models. This is summarized in Chap. 2 in the extensions along four directions. In particular, this theory is generalized to society diagnosis, i.e., to distinguish the guilty persons from the innocent and the unintentional liars by analysing their mutual testimonies. Some of these models are studied in detail as examples to show the scheme of proofs. Further, a simple analog circuit is taken as an example to illustrate how the system diagnosis theory has been applied to analog networks in recent years, and this will inspire people to apply this theory to the problem of multiple-fault diagnosis in logical circuits as well.

Design for testability is a hot project in the testing field, which can alleviate test generation and hence save computer time. But how to define the testability measure is still a problem. Chapter 3 lists the postulates for an ideal definition of testability measure and gives an evaluation of some existing definitions. It then introduces the synthesis problem for testability, which is more important in the designing phase. The automatic design for testability via testability measures is studied in detail at gate level, and the design at module level is also outlined.

The NMR (N Modular Redundancy) technique is widely used in fault-tolerant design. A new technique, NMRC (NMR with comparing), is introduced in Chap. 4, which contains the other NMR techniques as its special cases. Three optimal designs (Vertex-Edge, Edge-Vertex and Tm optimal) are studied in detail.

In recent years, multiprocessor systems and computer networks have become widely used, in which processors or computers are connected by buses with or without common memories, dedicated communication channels, crossbar networks or switching networks; their reliability problem should be well considered.

Chapter 5 studies the fault tolerance of an interconnection switching network formed by β-elements and other elements. Two special types, i.e., ISE and $C^n_{1,t}$, which provide the dynamic full access property, are taken into account. The number of faulty elements tolerated by the network can be increased by modifying the design without any penalty of additional hardware or time-delay. In addition an RFT network which provides the full access property and tolerates a larger class of fault models is also studied.

For multibus-multiprocessor systems, there seems to be lacking a unified mathematical tool which can be used to study them in depth. In Chap. 6, hypergraphs are used to represent multibus systems, so that the fault tolerance problem of the system corresponds to the connectivity problem of a hypergraph. An important inequality for the connectivities is proved and the best connectivity problem is solved with the aid of Balanced Incomplete Block (BIB) designs of combinatorics. This BIB design is generalized to Weak Balanced Incomplete Block designs (WBIB), which are new in combinatorics. Thus many optimal fault-tolerant system designs are obtained which are better than the previous ones. Based on this theory, a lot of work could be done in the future; this is laid before the reader.

TMR (Triple Modular Redundancy) system are widely used to achieve fault tolerance for single faults. If single faults occur successively in a system, there exists a fault-tolerant center with the fault-tolerant degree which is the maximum number of sequential faults tolerated. Hence the sufficient conditions of the optimal design for sequential faults are introduced.

References are listed, and some long proofs as well as tedious techniques are moved to the appendices after the chapters.

All the chapters are independent, and each chapter is self-contained, so the reader can read the book in any order. But it is necessary to have some basic knowledge of boolean algebra, graph theory, linear integer programming, testing and fault tolerance. Of course the reader should also be acquainted with the ideas of complexity of algorithms and NP-completeness.

This book can be regarded as the summary of the work done during the last ten years in the research group on fault tolerance in Chongqing University, the People's Republic of China. This material had been delivered twice as lectures in Chongqing University as a seminar for graduate students and also in Karlsruhe University, the Federal Republic of Germany, as a 24-hour course. This book was written based on these experiences.

The author is grateful to Prof. W. Görke and Prof. G. Hotz, who read the manuscript carefully and made some important suggestions. Also thanks are given to Dr. M. Marhöfer for his help in handling the manuscript. The author heartily thanks all colleagues in his research group; without their hard work and their original papers this book would have been impossible. Special thanks are given to Mr. Y.N. Chen who wrote the first drafts of Chap. 4 and Chap. 7, Dr. R. Yao, who wrote the final part of Chap. 6 and also Mr. F.J. Li, Mr. L.D. Zhou and Mr. Y.D. Shen for their typing. Special thanks should be given to my wife Prof. Heyue Wang for her typing the most part of this book.

Since fault tolerance is a wide research area, a lot of important contributions could not be included in one short book. Besides, there may be some mistakes or deficiencies in the book which have not been discovered by the author. If the reader would care to point these out, the author would appreciate it very much, and will make corrections in a revised version.

September 1991
Chongqing

Tinghuai Chen

Contents

Chapter 1
Four-Valued Logic and Its Applications

Four-valued logic is generalized from the traditional binary logic in order to describe the dynamic logical behavior of objective phenomena. We are going to study its mathematical features, especially the so-called STAR algorithm, the technique for deriving its component expressions, then to show how it can be applied to the static and dynamic test generations for both combinational and sequential circuits as well as to transitional logics.

1.1 Introduction

The traditional binary logic B_2 is well known to every computer scientist. It contains two logic states, "1" and "0" which can be used to describe the static state of objective phenomena, for example, the "true" or "false" state of a proposition, "high" or "low" voltage of a wire and "normal" or "faulty" state of a piece of equipment. But "steady" is only so in a relative sense. All objects are absolutely in a moving, changing, or transition state. In order to describe these transitional phenomena the traditional binary logic B_2 should be extended to some new logic.

In 1965 Roth proposed his well known D algorithm to generate tests for a combinational logical circuit, in which each wire is associated with one of the four values, $0, I, D$ and $\overline{D}$; where 0 or I denotes the state that this wire takes a steady logic value "0" or "1" in the common sense; D means this wire takes the logical value "1" when the circuit is normal and value "0" when a fault occurs in the circuit; conversely $\overline{D}$ means it takes "0" when the circuit is normal and "1" when a fault occurs. Using this system and a table driven technique, Roth successfully derived tests of a circuit but did not take into account the mathematical foundation of the system.

We are going to study this kind of four-valued logic from an abstract point of view and find out some other potential applications of this logic.

Since four-valued logic B_4 is generalized from B_2, we might seek inspira-

tion by reviewing how complex number theory is generalized from real number theory.

1. **Definition:** A complex number $\underline{a}$ is defined by an ordered pair of real numbers (a_1,a_2). The addition and dot multiplication are operated in component-wise

$$\underline{a} + \underline{b} = (a_1+b_1, a_2+b_2), \qquad \underline{a}\cdot\underline{b} = (a_1\cdot b_1, a_2\cdot b_2).$$

2. **Vector form** Each complex number can be expressed by a vector form:
$\underline{a} = a_1 + a_2\cdot i$. and this form is useful in computation.

3. **Components** It is important to have a technique to derive the real part $R(\underline{a}) = a_1$ and the imaginary part $I(\underline{a}) = a_2$ when $\underline{a}$ is a complex function.

Similarly we can define B_4 in terms of B_2, find the vector form and then propose a technique to derive the expressions of the components, which is called the STAR algorithm. This theory will be introduced in Sects. 1.2-1.5, while the applications will be handled in Sects.1.6-1.10.

1.2 Mathematical Basis

1.2.1 Four-Valued Boolean Algebra B_4

Let $B_2 = \langle B_2, "\cdot", "+", "-"; 0, 1\rangle$
be the binary logic B_2 with three logical operations "$\cdot$", "+" and "$-$" over two logical elements 0 and 1.

Definition 1.1 The four-valued logic B_4 is defined to be the direct (cartesian) product of B_2, i.e.

$$B_4 = B_2 \times B_2.$$

From discrete mathematics, B_4 is also a boolean algebra, and if $x, y \in B_4$, then x and y can be expressed by ordered pairs

$$x = (x_1,x_2) \qquad y = (y_1,y_2),$$

where x_1, x_2, y_1 and $y_2 \in B_2$ and the logical operations are performed component by component, i.e.

$$x\cdot y = (x_1\cdot y_1,\ x_2\cdot y_2),$$
$$x + y = (x_1 + y_1,\ x_2 + y_2),$$
$$\overline{x} = (\overline{x_1},\overline{x_2}).$$

Since B_2 has only two elements, the four elements of B_4 are simply

$$\theta = (0,0)$$
$$I = (1,1)$$
$$D = (1,0)$$
$$\overline{D} = (0,1)$$

and the operations are shown in Table 1.1,

Table 1.1

x	$\bar{x}$
θ	I
I	θ
D	$\bar{D}$
$\bar{D}$	D

$\cdot$	θ	I	D	$\bar{D}$
θ	θ	θ	θ	θ
I	θ	I	D	$\bar{D}$
D	0	D	D	0
$\bar{D}$	0	$\bar{D}$	0	$\bar{D}$

+	θ	I	D	$\bar{D}$
θ	θ	I	D	$\bar{D}$
I	I	I	I	I
D	D	I	D	I
$\bar{D}$	$\bar{D}$	I	I	$\bar{D}$

1.2.2 Boolean Expressions

Definition 1.2 Boolean expressions are defined recursively by the following

1) Constants '1' and '0' are expressions;
2) Variables are expressions;
3) The operations '+', '$\cdot$' and '$-$' on expressions finite times are expressions;
4) only those defined in 1), 2) and 3) are expressions.

Boolean expressions can be used to describe the structures of combinational logical circuits and are denoted by $A(x_1,\cdots,x_n)$, $B(x_1,\cdots,x_n)$ and so on.

Definition 1.3 Two expressions A and B are equivalent, denoted by

$$A(x_1,\cdots,x_n) \doteq B(x_1,\cdots,x_n),$$

if one of them can be transformed into the other by using the boolean identities finitely many times.

The following theorem can be found in any book on discrete mathematics.

Theorem 1.1 $A(x_1,\cdots,x_n) \doteq B(x_1,\cdots,x_n)$ iff $A(x_1,\cdots,x_n) = B(x_1,\cdots,x_n)$ for $\forall x_i \in \mathfrak{B}$, where $\mathfrak{B}$ is any boolean algebra.

Corollary 1.1 $A(x_1,\cdots,x_n) = B(x_1,\cdots,x_n)$ for $\forall\, x_i \in B_4$ iff $A(x_1,\cdots,x_n) = B(x_1,\cdots,x_n)$ for $\forall x_i \in \mathfrak{B}$ (in a special case, $\mathfrak{B}$ is simply B_2).

Example 1.1 Absorption Law: If $x + x\cdot y = x$ holds for $\forall\, x, y \in B_2$, then it also holds for $x, y \in B_4$. Therefore we have identities

$$D + D\cdot y = D, \quad \text{and} \quad x + x\cdot D = x.$$

1.2.3 Mapping $B_4^n \to B_4$ and Boolean Functions

Let $F(x_1,\cdots,x_n)$ be the mapping $B_4^n \to B_4$. Since there are 4^n points in B_4^n, each of them can be associated with one of the four values in B_4, and the total number of mappings is $|\{F(x_1,\cdots,x_n)\}| = 4^{4^n}$.

If $f(x_1,\cdots,x_n)$ is any boolean expression, when $\forall\, x_i \in B_4$, then $f \in B_4$. $f(x_1,\cdots,x_n)$ is a special mapping $B_4^n \to B_4$, called the boolean function, which can be used to describe the input-output behavior of a combinational logical

circuit. We know that the number of boolean expressions is 2^{2^n}, and hence

$$|\{f(x_1,\cdots,x_n)\}| = 2^{2^n}.$$

Therefore the set of boolean functions is only a subset of the mappings $B_4^n \to B_4$.

In order to study the property of these mappings and boolean functions, it is adequate to find some better representations for B_4.

1.2.4 Vector Form

Definition 1.4

For any $x \in B_4$ there exist four variables x^0, x^1, x^* and $\overline{x^*} \in B_2$, such that

$$\begin{aligned}
x = \theta \text{ in } B_4 \quad &\text{iff} \quad x^0 = 1 \text{ and } x^1 = x^* = \overline{x^*} = 0 \text{ in } B_2,\\
x = I \text{ in } B_4 \quad &\text{iff} \quad x^1 = 1 \text{ and } x^0 = x^* = \overline{x^*} = 0 \text{ in } B_2,\\
x = D \text{ in } B_4 \quad &\text{iff} \quad x^* = 1 \text{ and } x^0 = x^1 = \overline{x^*} = 0 \text{ in } B_2,\\
x = \overline{D} \text{ in } B_4 \quad &\text{iff} \quad \overline{x^*} = 1 \text{ and } x^0 = x^1 = x^* = 0 \text{ in } B_2. \qquad (1.1)
\end{aligned}$$

then $x \in B_4$ is called a **vector** and x^0, x^1, x^* and $\bar{x}^* \in B_2$ its **components**.

Obviously these components satisfy

Completeness $x^0 + x^1 + x^* + \overline{x^*} = 1$ and (1.2)

Orthogonality $x^i \cdot x^j = 0$ for $i \neq j$ and $x^i, x^j \in \{x^0, x^1, x^*, \overline{x^*}\}$. (1.3)

Since $B_2 \subset B_4$, the following expression is legal and is equal to x in any case:

$$x = x^0 \cdot \theta + x^1 \cdot I + x^* \cdot D + \overline{x^*} \cdot \overline{D}. \qquad (1.4)$$

Definition 1.5 Formula (1.4) is called the **vector form** of x. If the coefficients in (1.4) satisfy completeness and orthogonality, (1.4) is said to be **normalized**.

For the non-normalized vector form we have the following

Theorem 1.2 If vector form $x = p \cdot \theta + q \cdot I + r \cdot D + s \cdot \overline{D}$ is non-normalized, then the vector form $x = P \cdot \theta + Q \cdot I + R \cdot D + S \cdot \overline{D}$, where

$$P = \overline{q} \cdot \overline{r} \cdot \overline{s}, \quad Q = q + r \cdot s, \quad R = \overline{q} \cdot r \cdot \overline{s}, \quad S = \overline{q} \cdot \overline{r} \cdot s, \qquad (1.5)$$

is the corresponding normalized vector form.

It is easy to check the completeness and the orthogonality of this coefficient set {P,Q,R,S} and to check the equality of these vector forms.

Example 1.2 Let $x = a \cdot D + b \cdot \overline{D}$. Here $p = q = 0$, $r = a$, $s = b$. Using expression (1.5) to normalize it, we have

$$P = \overline{a} \cdot \overline{b}, \qquad Q = a \cdot b, \qquad R = a \cdot \overline{b}, \qquad S = \overline{a} \cdot b.$$

Thus the normalized form is $x = \overline{a} \cdot \overline{b} \cdot \theta + a \cdot b \cdot I + a \cdot \overline{b} \cdot D + \overline{a} \cdot b \cdot \overline{D}$.

1.2.5 Canonical Forms

Let F be a mapping $B_4^n \to B_4$. We are going to find a representation form for F.

A point of the space B_4^n corresponds to an assignment of one of the values θ, I, D, and $\overline{D}$ to the input variable x_i, i = 1,···,n, which is equivalent to $x_i^0 = 1$, $x_i^1 = 1$, $x_i^* = 1$ or $\overline{x_i^*} = 1$ respectively and i = 1,···,n. In other words this space point can be represented by $x_1^{j_1}\cdots x_1^{j_n} = 1$, where $j_1,\cdots,j_n \in \{0,1,*,\overline{*}\}$. This space point is mapped by F to one of the values θ, I, D, and $\overline{D}$. Thus F can be represented by the following logical sum

$$F = (\sum x_1^{p_1}\cdots x_n^{p_n})\theta + (\sum x_1^{q_1}\cdots x_n^{q_n})I + (\sum x_1^{r_1}\cdots x_n^{r_n})D + (\sum x_1^{s_1}\cdots x_n^{s_n})\overline{D}$$

The first sum containing the common factor θ can be ignored, and the second sum can be divided into two sums on account of $I = D + \overline{D}$. Hence we have

$$F = (\sum x_1^{k_1}\cdots x_n^{k_n})D + (\sum x_1^{l_1}\cdots x_n^{l_n})\overline{D} \qquad (1.6)$$

where the superscripts k_i, $l_i \in \{0,1,*,\overline{*}\}$ for i = 1,···,n and the first Σ take over all the inputs which cause F = 1 and D, the second Σ over all the inputs which cause F = 1 and $\overline{D}$.

Definition 1.6 Formula (1.6) is called the **canonical sum-of product form** of the mapping F, and each term in (1.6) is a **minterm.**

We can prove that every mapping $B_4^n \to B_4$ is uniquely expressed by its canonical sum-of-product form. Using this canonical form we can also prove that the set of all mappings $B_4^n \to B_4$ forms a boolean algebra B_{4^n} and hence all the boolean identities are applicable. Besides, the set of all the boolean functions of n variables forms a sub-boolean algebra $B_{2^{2^n}}$. We leave these proofs to the readers as exercises.

1.2.6 Expressions For Boolean Functions

Suppose $$x = x^0\cdot\theta + x^1\cdot I + x^*\cdot D + \overline{x^*}\cdot\overline{D}$$

and $$y = y^0\cdot\theta + y^1\cdot I + y^*\cdot D + \overline{y^*}\cdot\overline{D}$$

are normalized vector forms which can be transformed into canonical forms

$$x = (x^1 + x^*)\cdot D + (x^1 + \overline{x^*})\cdot\overline{D}$$

$$y = (y^1 + y^*)\cdot D + (y^1 + \overline{y^*})\cdot\overline{D}$$

Then the AND of x and y by Table 1.1 is

$$x\cdot y = (x^1 + x^*)\cdot(y^1 + y^*)\cdot D + (x^1 + \overline{x^*})\cdot(y^1 + \overline{y^*})\cdot\overline{D}$$

$$= (x^1y^1 + x^1y^* + x^*y^1 + x^*y^*)\cdot D + (x^1\overline{y^*} + x^1\overline{y^*} + \overline{x^*}y^1 + x^*\overline{y^*})\cdot\overline{D} .$$

Using the normalization formula (1.5) and after simplification we have the final normalized vector form

$$x\cdot y = (x^o + y^o + x^*\overline{y^*} + \overline{x^*}y^*)\cdot\theta + x^1y^1\cdot I \qquad (1.7)$$
$$+ (x^*y^1 + x^1y^* + x^*y^*)\cdot D + (\overline{x^*}y^1 + x^1\overline{y^*} + \overline{x^*}\overline{y^*})\cdot\overline{D}.$$

Similarly the OR of x and y is

$$x + y = x^oy^o\theta + (x^1 + y^1 + x^*\overline{y^*} + \overline{x^*}y^*)\cdot I \qquad (1.8)$$
$$+ (x^*y^o + x^oy^* + x^*y^*)\cdot D + (\overline{x^*}y^o + x^o\overline{y^*} + \overline{x^*}\overline{y^*})\cdot\overline{D}.$$

The NOT of x is $\overline{x} = x^1\cdot\theta + x^o\cdot I + \overline{x^*}\cdot D + x^*\cdot\overline{D}$ (1.9)

From the vector forms (1.7), (1.8) and (1.9) we can list out their corresponding component forms as follows.

Theorem 1.3 The components of an AND function are

$$\begin{aligned} (x\cdot y)^o &= x^o + y^o + x^*\overline{y^*} + \overline{x^*}y^* \\ (x\cdot y)^1 &= x^1y^1 \\ (x\cdot y)^* &= x^*y^1 + x^1y^* + x^*y^* \\ \overline{(x\cdot y)^*} &= \overline{x^*}y^1 + x^1\overline{y^*} + \overline{x^*}\overline{y^*} \end{aligned} \qquad (1.10)$$

Theorem 1.4 The components of an OR function are

$$\begin{aligned} (x + y)^o &= x^oy^o \\ (x + y)^1 &= x^1 + y^1 + x^*\overline{y^*} + \overline{x^*}y^* \\ (x + y)^* &= x^*y^o + x^oy^* + x^*y^* \\ \overline{(x + y)^*} &= \overline{x^*}y^o + x^o\overline{y^*} + \overline{x^*}\overline{y^*} \end{aligned} \qquad (1.11)$$

Theorem 1.5 The components of a NOT function are

$$\begin{aligned} (\overline{x})^o &= x^1 \\ (\overline{x})^1 &= x^o \\ (\overline{x})^* &= \overline{x^*} \\ \overline{(\overline{x})^*} &= x^* \end{aligned} \qquad (1.12)$$

Since $(\overline{x})^* = \overline{x^*}$, we can take off the parenthesis, and that is why we use the notation " $\overline{\ }^*$ " for the fourth component of a vector. In order to memorize these formulae, we can interpret them as follows. x·y, x+y, and $\overline{x}$ are outputs of the AND, OR, NOT gates respectively with inputs x and y. w^o, w^1, w^*, and $\overline{w^*}$ can be regarded as the voltage of a wire w such that, say,

w^o: "low" voltage ___ , w^1: "high" voltage ‾‾‾ ,

w^*: "step-down transition" ‾_ , $\overline{w^*}$: "step-up transition" _/‾ .

Thus the third formula of (1.10) can be so interpreted that the output of the AND gate is ‾_ if and only if one of the inputs x or y is ‾_, while the other is at high voltage or both of them are ‾_ .

STAR Algorithm The technique for deriving the expressions of the components of a given boolean function is called the **STAR algorithm**. Uaually there are three methods. Since a boolean function is formed by applying the operations "$\cdot$", "$+$", "$\bar{}$" finitely many times, its components can be derived by using formulae (1.10), (1.11), (1.12) iteratively according to its structure. (This is the first method for deriving the expressions of the components.)

In (1.10), (1.11), (1.12) the formulae for $(\ )^*$ and $\overline{(\ )}^*$ are symmetric, that is, we can obtain one from the other by interchanging all "*"s and "$\bar{}^*$"s.

Theorem 1.6 (Symmetry)

If $F^* = \varphi(0,1,^*,\bar{}^*)$, then $\bar{F}^* = \varphi(0,1,\bar{}^*,^*)$ and vice versa.

Example 1.3 For the boolean function $F(x_1,x_2)$ shown in Fig. 1.1 find the formula for F^* and $\bar{F}^*$.

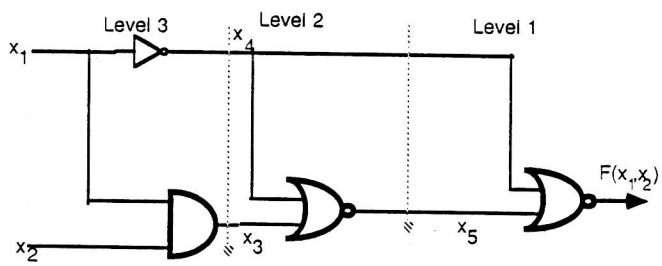

Fig. 1.1 Example

At level 1

$$F^* = (\overline{x_4 + x_5})^* = (\bar{x}_4 \cdot \bar{x}_5)^* = \bar{x}_4^* \bar{x}_5^1 + \bar{x}_4^1 \bar{x}_5^* + \bar{x}_4^* \bar{x}_5^*.$$

At level 2, $x_5 = \overline{x_3 + x_4}$,

$$F^* = \bar{x}_4^*(x_3 + x_4)^1 + \bar{x}_4^1(x_3 + x_4)^* + \bar{x}_4^*(x_3 + x_4)^*$$

$$= \bar{x}_4^*(x_3^1 + x_4^1 + x_3^* \bar{x}_4^* + \bar{x}_3^* x_4^*) + x_4^0(x_3^* x_4^0 + x_3^0 x_4^* + x_3^* x_4^*)$$

$$+ \bar{x}_4^*(x_3^* x_4^0 + x_3^0 x_4^* + x_3^* x_4^*).$$

By orthogonality $x_4^1 x_4^* = x_4^* x_4^* = x_4^0 x_4^1 = x_4^0 x_4^* = 0$,

$$F^* = x_3^1 x_4^* + x_3^* x_4^* + x_3^* x_4^0.$$

At level 3 $x_4 = \bar{x}_1$, $x_3 = x_1 x_2$,

$$F^* = (x_1^1 x_2^*)x_1^* + (x_1^* x_2^1 + x_1^1 x_2^* + x_1^* x_2^*)x_1^*$$

$$+ (x_1^* x_2^1 + x_1^1 x_2^* + x_1^* x_2^*)x_1^1$$