# A Study of Fault Detection Problems in Self-Checking Circuits

by

PANG Cho-Wai, Joseph, BEng(Hons)

A thesis submitted for the Degree of Master of Philosophy
in the Department of Electronic Engineering
of The Hong Kong Polytechnic University

Department of Electronic Engineering
The Hong Kong Polytechnic University

October 1997

# Abstract

The problems of self-checking circuits under different fault models such as multiple fault, open and bridging fault models were thoroughly examined and addressed. Various test generation and design-for-testability methods were investigated in order to find efficient solutions to tackle the problems. In particular, the following were achieved:

- The multiple stuck-at fault coverage problem was addressed. Off-line multiple stuck-at fault detection method was studied. In this study, a multiple stuck-at fault detection algorithm was developed for two-rail and parity checkers. The algorithm has been proved to achieve 100% multiple stuck-at fault coverage and requires less test vectors comparing to previously proposed method.
- The open and bridging fault coverage problems in CMOS circuits were studied, with particular emphasis on the bridging fault detection problem. Stuck-at fault model is insufficient and inadequate to model bridging faults in CMOS circuits. The detection of bridging faults by voltage testing is a very complicated task. We integrated the Iddq testing technique in the self-checking system and applied the test generation method to the two-rail checkers for bridging fault detection. By combining the multiple stuck-at fault detection and Iddq testing, the test quality can be greatly increased.
- The structures of domino-CMOS logic circuits are more testable than their static family for transistor stuck-open and stuck-on faults. However, the dynamic nature of the domino circuit structure prevents effective application of Iddq testing. Modification of domino-CMOS logic circuits was proposed to enhance the overall

testability of both voltage and Iddq testing. Furthermore, the extra cost of hardware is also very low.

- Another serious problem of the self-checking circuits is the misinterpretation of the output of the functional block due to the effect of bridging fault. In our earlier work, we focused on the use of built-in current sensor to detect bridging faults. However, Iddq testing is relatively slow in detection speed and suffers from low current resolution. The detection problem is mainly due to the occurrence of intermediate voltage at the fault site. In this respect, a sensing circuit called built-in intermediate voltage sensor(BIVS) was proposed to detect the intermediate voltage. Detailed analysis shows that the sensing circuit can achieve the self-checking requirements. So that it is suitable for on-line testing applications. An integration of BIVS and self-checking system is further proposed as an application example.

# Table of Contents

# List of Figures

# List of Tables

# List of Publications

**JOURNAL PAPER:**

J. C. W. Pang, M. W. T. Wong, and Y. S. Lee, "An efficient test generation for multiple fault coverage of two-rail checkers", International Journal of Electronics, 83, 6, Dec. 1997, pp. 837-848.

**CONFERENCE PAPERS:**

J. C. W. Pang, M. W. T. Wong, and Y. S. Lee, "An efficient test generation for multiple fault coverage in two-rail checkers", Proc. 2nd IEEE Int'l On-Line Testing Workshop, France, pp. 118-123, July 8-10, 1996.

J. C. W. Pang, M. W. T. Wong, and Y. S. Lee, "Hybrid test generation approach for multiple fault detection of TSC system implemented with CMOS two-rail checker", 7th Int'l Symp. On IC Technology, Systems and Applications(ISIC-97), Singapore, Sept. 10-12, 1997.

J. C. W. Pang, M. W. T. Wong, and Y. S. Lee, "Design and implementation of strongly code-disjoint CMOS built-in intermediate voltage sensor for self-checking circuits", accepted for presentation in Asian Test Symposium 1997(ATS'97), Japan, Nov 17-19. 1997.

J. C. W. Pang, M. W. T. Wong, and Y. S. Lee, "On testability analysis of domino-CMOS logic circuits", accepted for presentation in 7th Symp. on Fault Tolerant Computing(CFTC-7), China, Dec. 11-12, 1997.

# Acknowledgements

I am indebted to express my sincere gratitude to everyone who giving up their time and support to this research.

Thanks to Dr. Mike W. T. Wong, my chief supervisor, for his valuable advice and helpful assistance. Without his intensive knowledge, initiative and great insight, the goal of this research work would not be achieved successfully. I have got great benefit by his guidance, supervision and encouragement. Both of my knowledge and character are improved.

I wish also to thank Professor Y. S. Lee, my second supervisor, for his ever-ready support and many thoughtful suggestions. His enthusiasm and experience enriched my research work.

Finally, thanks to the Hong Kong Polytechnic Research Committee for providing financial support and without which support, the research would be nothing more than an idea.

# Chapter 1

# Introduction

## 1.1 INTRODUCTION

Digital systems have been widely adopted in the past decades and become irreplaceable in modern society. Computing systems, communication networks, control systems are made possible by the advance in microelectronics technology such that the performance of these systems is greatly increased. As our real life becomes inseparable from them, these systems should be built with as high quality as possible.

A global concept called dependability which subsumes the usual attributes of reliability, availability, safety, maintainability, performability and testability describes the quality of a system [PRAD 86][JOHN 89]. High dependability is always desired. However, physical defects, imperfection or flaws may occur in the system during manufacturing phase, or even in operating phase. Periodic off-line testing may be used to discover any defects before shipping the products to the customer, but it is not effective to detect the temporary faults. Once the product is working in normal operation, errors produced by hardware faults will remain undetected until the test phase. Also, off-line testing is not effective against temporary faults. Those factors will affect the quality of a system. Furthermore, design error is also the main source of error. In order to build a highly dependable system, fault-tolerant(FT) design techniques have been proposed. Two major goals of FT system are: 1) the system is capable of producing correct output under the occurrence of some restricted error types, and 2) the system is capable of producing extra signal to indicate the error under the occurrence of some restricted error types. Consequently, the availability and reliability can be significantly increased.

Nowadays, FT systems have been widely used to provide highly reliable and safe-critical applications. However, there are some drawback from the existing design techniques. For example, in a triple modular redundancy(TMR) system [PRAD 86][JOHN 89], a voter performs majority vote on the outputs of three identical systems. Any single system error will be masked at the output of the voter. However, the voter is not checked. Also, large cost due to additional hardware is required. Generally, the problem of "checking the checker" becomes the major difficulty in FT design.

Based on this observation, the development of totally self-checking(TSC) goal is motivated. That is, the first erroneous output of the functional block or checking circuit(checker) must be a non-codeword. Basically, in TSC system, the functional block is monitored by a checker concurrently. The necessary mathematical properties that the circuits must be verified in order to achieve the TSC goal were firstly introduced by [CART 68]. Later, the largest class of TSC functional circuits were defined in [ANDE 71][SMIT 78]. More recently, the largest class of checkers necessary to ensure the TSC goal has been defined by [NICO 88].

Synthesis of self-checking circuits is a major topic in FT field. The design methods of the checkers have been intensively studied by [ADER 71][ADER 73][REDD 74][MARO 78][HUGH 84][KHAK 84][FUJI 87][MIN 88][LO 90][PASC 90][FUJI 91][OZGU 91][TAO 92][RAO 93][BURN 94][METR 94a][REDD 94][DIMA 95]. Design methodologies for functional circuits can be found in [HIRA 91][BUSA 94][SAPO 96]. In practice, the effectiveness of TSC property is dependent on the predetermined fault model. That is, all possible faults in the predetermined fault model must satisfy the mathematical properties of TSC goal. In most of the studies, the fault model used is the single stuck-at fault model. Several design methods for different checkers based on the PLA implementation have been developed [MIN 88][HATA 91][TAO 92]. Furthermore, it has been showed that dynamic CMOS and NMOS logic circuits are more applicable to realise self-checking circuit(SCC) designs rather than fully static CMOS family under the realistic fault model such as open fault [JHA 84][JHA 90][CHEE 92]. Generally, there is still no design methodology to achieve high realistic fault coverage in SCCs.

## 1.2 OUTLINE AND OBJECTIVES OF THE THESIS

In our study, two major areas have been identified which are important in SCC designs. They are, namely, the multiple fault problem and the realistic fault coverage problem. Firstly, multiple fault is the concurrently occurrence of two or more than two faults in the circuit. In the past decades, automatic test pattern generation(ATPG) algorithms have been developed based. These are on the single stuck-at fault model only because the computational complexity of multiple fault is very large [FUJI 85][ABRA 90]. Also, for on-line testing approach, single fault model is used because it is very hard to implement a TSC system under multiple fault assumption [NANY 88][NICO 89a]. However, many physical defects may only be modelled as multiple faults. Therefore, the ability to detect the multiple faults in early stage is highly desired.

Secondly, in order to enhance the quality of testing scheme, realistic fault detection must be considered. As reported in [WADS 78][GALI 80][LEVI 81][MALA 82][SHEN 88][SHER 88][STOR 90][FERG 91][STOR 91][LIDE 92][LEE 96a][LEE 96b][WEN 97], realistic faults such as bridging faults and open faults are hard to be detected by logic testing approach because they include the degradation in certain parameters such as voltage, current and timing. Furthermore, these faults may not result in logical error at the primary output node such that they are hard to be detected. Recently, the fault coverage problem of present SCCs have been investigated by [JHA 84][JHA 90][MILL 91][LIDE 92][JHA 93][NICO 94]. Different techniques to handle this problem were carried out by [JHA 84][NICO 91][CHEE 92][LO 95][METR 95][FAVA 96]. In particular, modification of the circuit layout may be required in their methods which may not easily apply to existing CAD tools. In our study, we have extended our study to realistic fault model and to explore methods to tackle the problem more effectively.

The main contents of this thesis are as follows:

- To review the concept of fault, error and failure and also the principle of coding theory are discussed. And to study the principles and concepts of self-checking properties as background knowledge to support the work to be presented.

- To study the multiple fault testability of existing self-checking circuits. Off-line multiple stuck-at fault detection methods were studied. In the following study, attention is focused on the following two important issues of existing self-checking circuits: 1) present TSC circuits are designed to achieve concurrent error detection of single fault only, but not multiple fault and 2) although off-line testing for multiple fault is very important, very little work has been done in this area. In this thesis, extensive coverage on the multiple fault detection problem of two-rail and parity checkers will be included. The fundamental benefit of multiple fault detection allows the selection of high quality circuits during manufacturing test, resulting in higher mean-time-to-failure.

- To study the realistic fault coverage problem in static CMOS self-checking circuits. Stuck-at fault model is insufficient and inadequate to model many CMOS faults. In this thesis, we are mainly concerned with the detection of bridging fault because this kind of fault is hard to be detected by logic testing. We shall integrate Iddq testing method with the logic testing approach. Based on this implementation, an example of detecting bridging fault in two-rail checkers will be used to show the effectiveness of our implementation.

- To improve the Iddq testability in domino-CMOS logic circuits. The structures of domino-CMOS logic circuits are more testable than their static family for transistor stuck-open and stuck-on faults. However, the inherent dynamic structure prevents the application of Iddq testing. Modifications of domino-CMOS logic circuits are proposed to enhance the overall testability of both logic and Iddq testing.

- To study the bridging fault detection problem for on-line testing application. In the previous work, we have focused on the use of built-in current sensor(BICS) to detect bridging faults. However, Iddq testing is relatively slow in detection speed and suffers from low current resolution. The detection problem is mainly due to the occurrence of intermediate voltage(indeterminate logic value) at the fault site. Problem arises especially when this effect appears at the input of the checker. The checker may fail to detect it. Alternative method called built-in intermediate voltage sensor(BIVS) to detect the intermediate voltage is proposed. Detailed analysis will be carried out to show the effectiveness of BIVS. The integration of BIVS and self-checking system is also proposed as an application example.

To summarise, the main aims of this research are: i) in-depth study of the fault coverage problems of SCCs under different fault models, and ii) design and implement effective solutions in terms of different detection and design-for-testability techniques for these problems.

# Chapter 2

# Background Theory

## 2.1 INTRODUCTION

In this chapter, we shall present the basic terminology and the definitions of several key concepts used in the fault-tolerant field as the background knowledge for this research work. It is imperative to state any technical terms used clearly because there are ambiguities arising from different interpretation of the same word. For example, [JOHN 89] has defined physical defects as faults while [PIES 95] has considered that a failure is a physical defect. This provides the basis for a good overall understanding of the causes of faults, the types of faults and the effects of the faults which are essential in this study. Furthermore, the important aspect of fault-tolerant computing field and the properties of error-detecting codes will be discussed.

## 2.2 FAULTS, ERRORS and FAILURES

Here we concern the definitions of faults, errors and failures in a digital system. The terminology presented here is based on [JOHN 89] whose definitions are widely accepted in the field.

**Definition 2.1.** A fault is a physical defect, imperfection, or flaw that occurs within some hardware component in a circuit. Examples of faults include bridging between two electrical conductors, a break or open in conductors, or physical flaws or imperfections in semiconductor devices such that the parameters are not within the specifications.

Faults can be classified as logical or parametric, and permanent or temporary. A logical fault is one that causes the logic function of a circuit element to be changed to some other logic function. The typical example of logical faults is logical stuck-at-$z$ fault (s-at-$z$). This fault model assumes that the logic value on an input or output line of a circuit element is set to logic $z$ permanently, independently of the input value applied to the faulty line. On the other hand, a parametric fault may result in degradation in circuit parameters, causing a change in magnitude of some factors such as signal timing, current, or voltage. Examples of these faults may include bridging faults and open faults.

Further classification of faults may include the time of duration. Permanent faults that always present and do not disappear, or change their nature during testing. These faults are mainly due to the imperfection in the manufacturing process. Temporary faults, which have been predominant in modern digital system, include two types of faults: transient faults and intermittent faults. Transient faults are faults which may appear once and may never appear again. It may be caused by power supply fluctuation, electromagnetic perturbations, temperature and humidity variation, radiation etc. Intermittent faults may appear, disappear and then reappear repeatedly. Hence, there are no reliable means of detecting their occurrences of temporary faults because they may disappear when a test is applied or may appear during system runtime. The early detection of temporary faults for the reliable operation of the system is very important, hence it motivates the development and implementation of fault-tolerant techniques to detect or tolerate the occurrence of temporary faults as well as permanent faults during the system's life cycle.

**Definition 2.2.** An error is the manifestation of a fault. In other words, it is the occurrence of an incorrect state at the observation nodes caused by faults within a circuit. For example, a fault occurs inside a circuit and is activated by suitable inputs, the effect may only be observed at the primary output of the circuit. If the observed output deviates from the normal one, an error in operation of the circuit is occurred.

**Definition 2.3.** A failure occurs in a digital system when the system performs one of its functions incorrectly. It is also considered as the under specifications of same functions in terms of quantity and quality.

It is clear that there are cause-and-effect relationship between faults, errors and failures. Faults, when activated, may result in an error. However, error may not occur if the fault is redundant or the effect of the activated fault is negligible. When a system experiences errors, it may be lead to failures. From another point of view, errors are the effect of faults, and finally, failures are the effect of errors.

## 2.3 FAULT MODELS

Throughout our research, it is necessary to pre-determine the kinds of faults that we are considering in the analysis. In practice, large numbers and types of probable defects can happen, in order to make the analysis more effective and more meaningful, only a subset of all possible faults will be considered. This is because the complexity of analysing multiple fault coverage is too high and some faults may not occur in a particular technology.

Fault model allows us to specifically define the types and the numbers of faults. In our study, we consider three primary fault models: 1) logical stuck-at fault model at gate-level , 2) stuck-open fault model, and 3) bridging fault model at the transistor-level.

### 2.3.1 Logical Stuck-at Fault Model

The most common fault model is the logical stuck-at fault model or simply the stuck-at fault model. This fault model includes two types of faults: 1) stuck-at 0 fault (s-at 0), and 2) stuck-at 1 fault (s-at 1). Stuck-at z fault assumes that the faulty circuit node is set to value logic z permanently, independent to the applied value on the node. This fault model is most popular because it is simple and easy to manipulate. Also, it can represent a large number of possible defects in the circuit. Figure 2.1 further illustrates the application of stuck-at fault model. For example, an input of an AND gate is stuck-at 1 (e.g. line A is stuck-at 1). It is easy to verify that when the input vector AB is 10 or 11, the fault-free gate produces correct output value 0 and 1 respectively.

However, when the stuck-at fault is activated, it produces incorrect output. Table 2.1 summaries the operation of a fault-free and faulty AND gate to demonstrate the stuck-at fault model. In this example, we also demonstrated how a fault is activated. A fault is considered as important when it can result in an error.

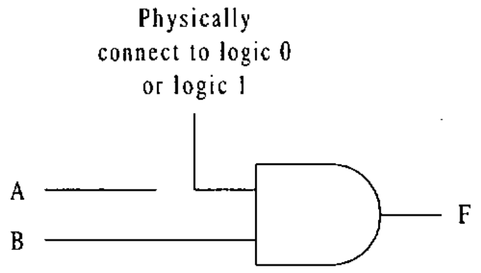


**Figure 2.1** Illustration of the application of the logical stuck-at fault model.

**Table 2.1** Outputs of a fault-free and a stuck-at 1 AND gate.

| Input test vector AB | fault-free output | faulty output |
|---|---|---|
| 00 | 0 | 0 |
| 01 | 0 | 1 |
| 10 | 0 | 0 |
| 11 | 1 | 1 |

Usually, in most test generation algorithms, testability analysis and fault-tolerant system designs, single s-at fault model is assumed. For a circuit with n nodes, it will have at most 2n number of s-at faults. However, there will be $3^n$-1 possible multiple s-at faults. The computational complexity of multiple stuck-at fault grows exponentially as the circuit size increases. On the other hand, it has been shown in

[ABRA 90] that test set for single s-at fault coverage also provides a very high fault coverage of multiple s-at fault as well. In general, it is restricted to single fault because it is very hard to design a fault-tolerant system that is capable of handling multiple faults [NICO 89].

As CMOS technology becomes the most popular VLSI design technology, it is necessary to verify the effectiveness of the s-at fault model versus the actual CMOS VLSI circuit faults. Stuck-at fault model is found to be insufficient and inadequate to model CMOS VLSI circuits faults [GALI 80][MALA 82][WADS 78]. Bridging and open faults are the most representative CMOS VLSI circuit faults, however, these two faults cannot be represented by stuck-at faults precisely. In order to model these faults accurately, it is necessary to extend our fault model to transistor-level as opposed to gate-level. Transistor-level fault model is sometimes known as switch-level fault model because a transistor is essentially performing a switching function.

### 2.3.2 Stuck-Open Fault Model

It might be thought that all the silicon chips that are processed using a set of correctly designed photomasks would work at the first time. Practically, this is not the case. Defects may occur at manufacturing stage or during the chip's life cycle. Stuck-open fault is a fault in which there is a break at the conductor. It might result in misalignment of different masks during VLSI manufacturing process, over-removal of polysilicon and metal during etching, failure in cutting oxide windows, etc. Examples of electromigration and corrosion of metal due to trapped moisture can result in stuck-open fault during the chip life-cycle. In general, stuck-open fault will result in high impedance at the fault site. It may turn the combinational circuit into a sequential one. An example of CMOS two-input NAND gate as shown in Figure 2.2 illustrates the behaviour of stuck-open fault. Assume that a break occurs between line A and gate node of transistor N2. This fault prevents the conducting of transistor N2 while input AB is set to 11. That is, output F is at high impedance state. As a result, the value of F is depending on the previous state. Table 2.2 shows the truth table for the two-input CMOS NAND gate for both the fault-free and faulty conditions. Based on this observation, two-pattern tests is required to detect stuck-open fault [GALI 80][MALA 82][WADS 78]. Based on this technique, firstly a suitable input pattern initialises the output of a gate to a known state. Then, another input pattern, which normally produces a complement output to the previous one, is applied. If the output of the gate does not toggle, stuck-open fault is detected.

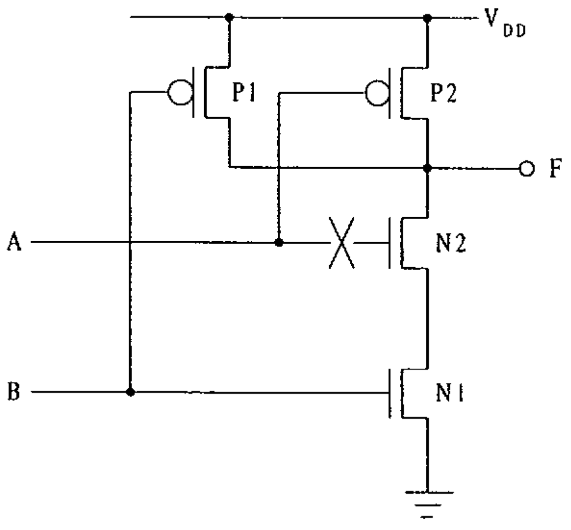


**Figure 2.2** CMOS two-input NAND gate.

**Table 2.2** Truth table for CMOS NAND gate.

| Input pattern AB | Fault-free output | Faulty output |
|---|---|---|
| 00 | 1 | 1 |
| 01 | 1 | 1 |
| 10 | 1 | 1 |
| 11 | 0 | Previous state |

### 2.3.3 Bridging Fault Model

Bridging fault is caused by an excess connection between two nodes. This fault is activated when the bridged two nodes are set at different logic values. The effects of bridging faults are dependent on the process technology used. For some technologies such as TTL and ECL, wired-AND or wired-OR model is valid since they are either logic high dominate or logic low dominate. However, it has been shown that these two models are inadequate for CMOS technology [GALI 80][LEVI 81][MALA 82][SHEN 88].

For CMOS logic gate, a logic high at the output of a gate means that there is at least one serial connection of PMOS transistor(s) connecting to the $V_{DD}$ node. On the other hand, a logic low at the output of another gate means that there is at least one serial connection of NMOS transistor(s) connecting to the ground node. If the two output lines are bridged, the output voltages of the two gates will depend on the resistance of the pull-up transistor(s), the resistance of the pull-down transistor(s) and the bridging resistance [LEE 96a][LEE 96b][STOR 90][TANG 95]. For example, let

A and B be the two bridged lines. Let the correct value of A be logic 0 and of B be logic 1 and let I be the indeterminate voltage. When the resistance of the bridging is increased, the current across the bridging branch will be decreased. Thus, the voltage at A will move from I to 0 and the voltage at B will move from I to 1. Hence, when the short resistance is increased, this effect is amplified. Based on the above observation, it can be seen that the resultant value of the two bridged lines can be any value between $V_{DD}$ and ground providing that voltage at line B is always larger than voltage at line A. Bridging faults can occur between any two electrical nodes in the circuit. In practice, the occurrence of bridging faults is dependent on the physical layout of the circuit. In most cases, intratransistor bridging faults and gate-level bridging faults are the most popular and are considered in detailed in our study.

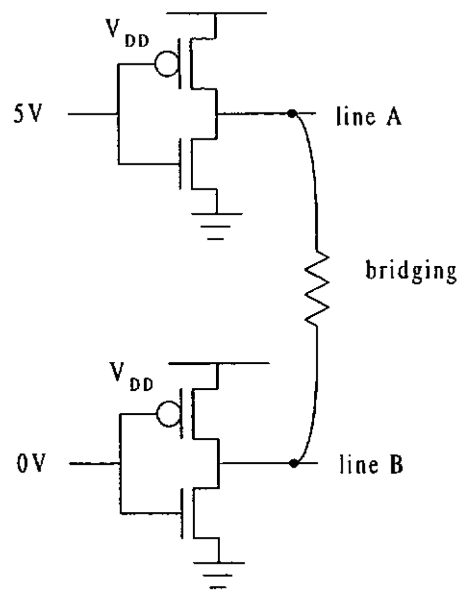


**Figure 2.3** Bridging fault between two outputs of inverters.

For most practical purposes, most faults resulting from physical defects can be modelled by stuck-at, stuck-open or bridging faults. However, not all faults can be modelled by these fault models, for example, such as crosspoint faults in PLA, delay faults etc. In this thesis, we shall consider stuck-at , stuck-open and bridging faults only.

## 2.4 BASIC PROPERTIES OF ERROR-DETECTING CODE

On-line testing schemes are usually based on redundancy techniques: hardware redundancy and information redundancy. Information redundancy is the addition of extra information by means of encoding of data with an error-detecting code(EDC) or an error-correcting code(ECC). Popular coding schemes such as parity, two-rail code, Berger code, Hamming code are used in information redundancy. Hardware redundancy is the addition of extra hardware for the purpose of either fault detection or fault tolerance. For self-checking designs, both redundancy techniques are implemented. Self-checking circuits(SCCs) are designed based on hardware encoding techniques such that the output is a kind of EDCs. Extra hardware is always required in order to achieve the totally self-checking(TSC) goal [BUSA 94][LALA 85]. In this section, we briefly introduce the basic properties of error-detecting code used in digital systems [PIES 95][PRAD 86].

For a combinational circuit of n input lines and m output lines, there will be $2^n$ number of input data and $2^m$ number of output data. In order to distinguish the status of the circuit easily, SCCs are implemented such that a subset of all possible output space *C*, called codewords, will be produced and will be considered as correct outputs during on-line normal operation. And non-codewords are considered as errors. Then, the status of the circuit can be identified. In general, the output of the SCCs is encoded to a kind of EDCs such as parity code, residue code, Berger code, m-out-of-n code, etc. We explain the properties of EDC as follow:

**Definition 2.4.** For any two codewords(say $C_1$ and $C_2$) in code space $C$, if the number of different bits between $C_1$ and $C_2$(Hamming distance) is equal to or larger than 2, $C$ is capable of detecting any single bit error. For example, given two odd parity codewords 0111 and 0010, differ in two positions. Therefore, the Hamming distance is 2. And clearly, if codeword 0111 is contaminated by a single bit error, it will be changed to an even parity word and error is detected.

**Definition 2.5.** Given two codewords $C_1$ and $C_2$ in code space $C$. We say that $C_1$ covers $C_2$ if and only if $C_1$ has 1s everywhere $C_2$ has 1s. If neither $C_1$ covers $C_2$ nor $C_2$ covers $C_1$, $C$ is called an unordered code. For unordered code, the minimum Hamming distance of the code is always larger than or equal to 2. That is, it can detect any single or unidirectional error. Therefore, Berger code is more effective in error detection than parity code.

**Definition 2.6.** A code is called systematic if the information bits and the check bits are separable. For example, Berger code is a kind of systematic code, while m-out-of-n is non-systematic code.

The selection of a EDC in a particular application may be dependent on the following factors:

1. In many applications, more than one output line can be affected by a fault inside the circuit. The circuit is modified such that any fault will produce single or unidirectional errors. So, unordered code may be suitable for this kind of applications.
2. Systematic code is convenient to use because the information is separable. The self-checking circuit is also easier to modify or design. For non-systematic code, extra decoding hardware is always necessary to produce the required outputs and the decoding hardware is not "protected" by any coding technique. On the other hand, many circuits such as PLA-based design circuits have shown the use of non-systematic code efficiently.
3. Since self-checking design always introduces extra hardware (for both the modification of the functional circuit and the checker), the circuit complexity and the speed degradation become important considerations.

The effectiveness of encoding the output of a digital circuit in a suitable EDC is highly dependent on the effect of the faults inside the circuit. As demonstrated in previous paragraphs, existing EDCs for digital systems are capable of detecting single or unidirectional bit errors, but not multiple random bit errors. That is, it is necessary to verify that all possible faults inside the circuit will only result in single and unidirectional bit errors. Otherwise, these faults may not be detected by the monitoring circuitry. Hence, a class of circuits called self-checking circuits are defined to achieve this requirement. Details of which will be introduced in the next chapter.

## 2.5 SUMMARY

In this section, we introduced several terminology in the fault-tolerant computing field. Relationships between faults, errors and failures were clearly defined. In practice, we shall consider the fault models and error models only. Usually, stuck-at fault model is adopted. However, stuck-at fault model is inadequate and insufficient to represent stuck-open and bridging faults in CMOS VLSI circuit. We have extended our consideration to include the more realistic faults. Finally, we have explained in detail the properties of error-detecting code. EDC plays an important role in fault-tolerant field. Equipped with this knowledge, we can define the error model more accurately and we can implement the fault-tolerant computing system more effectively.

# Chapter 3

# Self-Checking Circuits

## 3.1 INTRODUCTION

The use of self-checking system was motivated by the low reliability of the components in the past. As the design and manufacturing technology advances, component reliability has been significantly improved. On the other hand, the complexity of VLSI circuits grows quickly with the decreasing size of internal devices. However, as the complexity of circuit and system increases, the cost of testing such complicated system grows rapidly. Also, the systems are more and more prone to temporary faults [MASA 88]. These intermittent and transient faults are hard to detect by the traditional off-line testing approach. To meet the challenge, concurrent error detection(CED) by the self-checking design technique providing efficient on-line testing capability is required. The implementation of self-checking technique requires extra overhead. However, the gain in system quality and reliability outweighs the cost of additional circuitry.

The objective of designing self-checking circuits is to achieve the totally self-checking(TSC) goal, that is, the first erroneous output of the functional circuit must not belong to the output code. The necessary mathematical properties that the circuits must have in order to achieve the TSC goal was firstly introduced by [CART 68].

Later, the largest class of TSC functional circuits were defined in [ANDE 71][SMIT 78]. More recently, the largest class of checkers necessary to ensure the TSC goal was defined by [NICO 88]. Self-checking circuits now become an important topic in fault-tolerant field [ABRA 90][FUJI 85][JOHN 89][LALA 85].

In general, the basic structure of these circuits (as shown in Figure. 3.1) consists of two main blocks: a functional circuit and a checker. The output of the functional circuit, in fault-free operation belonging to a suitable error-detecting code, feeds the checker. The checker should be capable of detecting not only the presence of a non-code word at its inputs, but also faults in itself. In this chapter, the properties of self-checking systems will be introduced.

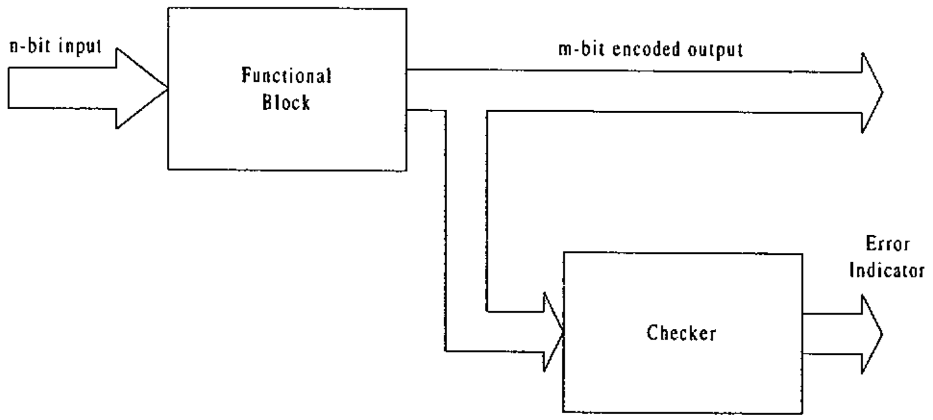


**Figure 3.1** Block diagram of a self-checking system.

## 3.2 SELF-CHECKING CIRCUITS

In this section, we introduce the necessary properties to implement the functional block such that the overall system can achieve the TSC goal. During normal operation, the output of the functional block belongs to a suitable EDC. A non-codeword at the output of the functional block indicates the presence of a fault. However, a fault may also result in incorrect codeword at the output. So, we need the following definitions to describe the manner in which self-checking circuits deal with faults.

**Definition 3.1.** A circuit is fault secure(FS)[ADER 71][ADER 73] with respect to a given fault set F if it never produces incorrect codeword output for any input for every fault f in F.

A FS circuit will never produce incorrect codeword output when a fault occurs. However, it cannot guarantee that the fault can be indicated by means of non-codeword output. Additional property is required.

**Definition 3.2.** A circuit is self-testing(ST)[CART 68] with respect to a given fault set F if it produces non-codeword output for at least one input for every fault f in F.

**Definition 3.3.** A circuit is totally self-checking(TSC)[ADER 71][ADER 73] if it is both self-testing and fault-secure for every fault f in F.

Existing self-checking circuit designs are based on the following fundamental assumption:

**Assumption 3.1.** At any time, only one fault can occur and the time interval between the occurrence of two faults is sufficient long such that all the required test vectors can be applied to the circuit. In other words, TSC circuits are defined respect to single fault assumption.

Apparently, a TSC functional circuit for a given fault set guarantees its correct operation in the presence of faults. Once the fault is activated and the effect is propagated to the output, it must be a non-codeword output. Designing of a circuit with TSC property is always desirable because every fault inside the circuit can be detected by codewords. However, it is hard to achieve TSC respect to some realistic fault models [JHA 84][JHA 90][JHA 93]. So, a more general definition was introduced.

**Definition 3.4.** A circuit is strongly fault secure(SFS)[SMIT 78] for a fault set F if and only if, for every fault f in F, either

1) the circuit is TSC when f occurs, or,
2) the circuit is still FS but not ST, and the resultant circuit is still SFS for the remaining fault in the fault set.

In other words, undetectable faults are allowed in a SFS circuit. However those faults will not result in incorrect codeword output.

The SFS circuits are the largest class of functional circuits that can achieve the TSC goal. A SFS circuit can be transformed to a TSC circuit if all the redundancy faults are removed. However, application of SFS property is quite complicated. It is necessary to verify that under the occurrence of all combinations of all possible undetectable faults, the circuit is SFS. [WANG 94] stated that it is difficult to achieve TSC goal without significant overhead in terms of hardware cost and extra delay.

## 3.3 SELF-TESTING CHECKERS

The function of the checker is to determine whether the output of the function block is a codeword or not. Generally, if the output is a codeword, 01 or 10 is generated by the checker output. For non-codewords, the checker will indicate 00 or 11 as error signal. Obviously, the checker generates two outputs that can overwhelm the problem of output line stuck-at fault. Also, two-rail encoded outputs can detect the wired-AND or wired-OR bridging between output lines that otherwise cannot be detected by identical outputs. So, for a checker, we have the following definitions.

**Definition 3.5.** A circuit is code-disjoint(CD)[ADER 71][ADER 73] if it always maps input code space to output code space and input non-code space to output non-code space under normal condition.

**Definition 3.6.** A self-testing checker[ADER 71][ADER 73] is both code-disjoint and self-testing.

In fault tolerant field, "checking the checker" is a universal problem. An ST checker can detect any error at its input, but must also be capable of detecting faults in itself. A TSC functional block monitored by an ST checker can achieve the TSC goal. Let us use the following two examples to demonstrate this point.

**Example 3.1.** If a fault occurs inside the TSC functional block, it will produce a non-codeword for at least one input vector. The checker detects the non-codeword and produces an error signal.

**Example 3.2.** If a fault occurs inside the ST checker, it will be detected by the codeword which is fed by the functional block.

Practically, it is hard to achieve that all faults inside the checker must be detectable. Also, it was thought that under the occurrence of some faults, the checker can still maintain CD property, but not ST. The desired function of a checker is not affected. So a more general property of a checker is defined as follow:

**Definition 3.7.** A checker is strongly code-disjoint(SCD)[NICO 88] for a fault set F if and only if, for every fault f in F, either

1) the circuit is ST and CD when f occurs, or
2) the circuit is still CD but not ST, and the resultant circuit is still SCD for the remaining fault in the fault set.

In other words, undetectable faults are allowed in an SCD checker. However, the checker is still capable of mapping non-codeword input to non-codeword output.

In both cases, FS property is not necessary for an ST or SCD checker. The ST property ensures that all internal faults will result in non-codeword output and the CD property ensures that all non-codewords at the input will be mapped to non-codewords at the output. [JHA 90] proposed that the output code space may be reduced to either codeword 01 or codeword 10 under the occurrence of the undetectable fault sequences. In a complex self-checking system, a final two-rail checker(TRC) is used to multiplex output signals of the checkers. Based on the observation, the final TRC may not receive sufficient codeword to fully exercise itself. It means that some faults may not be activated due to insufficient codewords. Therefor, FS property may be useful to the checker. However, [NICO 94] showed that the above observation will not happen for realistic fault models. The reduction in code space is not likely to occur. So, the final TRC can still be fully exercised. On the other hand, in such a complex system, it is necessary to verify that each subsystem can receive all required input tests [BOUD 91].

## 3.4 SUMMARY

In this section, we defined the necessary conditions to achieve the totally self-checking goal. Fault-secure and self-testing are the basic requirements that a TSC functional circuit has to meet. Furthermore, the functional circuit must be monitored by self-testing checker to achieve both self-testing and code-disjoint properties. This system is called totally self-checking system. In practice, under some realistic fault models, the circuit may not achieve totally self-checking property. Strongly fault-secure and strongly code-disjoint properties are defined such that redundancy faults are allowable. An SFS functional block monitored by an SCD checker can achieve the TSC goal.

# Chapter 4

# Analysis of Multiple Stuck-at Fault Coverage in SCCs

## 4.1 INTRODUCTION

In this chapter, we study the multiple stuck-at(s-at) fault coverage in self-checking circuits(SCCs) in general and the self-testing checkers(STCs) in particular. For many years, SCCs are designed based on the single fault assumption as stated in Chapter 3. It is assumed that fault occurs once at a time and before the occurrence of next fault, enough codewords are presented at the input of the SCCs(both the functional block and the checker). From another point of view, SCCs are highly testable circuits because there are no redundancy faults. Self-testing(ST) property ensures that all faults will result in non-codeword for at least one input. Therefore, all single faults in the predefined fault set must be testable.

As the complexity of VLSI circuits increases rapidly, the cost of testing such complicated circuits increases. However, testing must be carried out before shipping the products to customers. Since a large number of physical defects can happen on the VLSI chips, these fabrication defects may be modelled as multiple faults. In the past few decades, test pattern generation algorithms are mostly focused on single s-at fault model [ABRA 90][FUJI 85][JOHN 89][LALA 85] because the computational complexity of multiple fault detection issue is very high. For a circuit with n lines, it will have at most 2n number of single s-at faults. However, there will be up to $3^n$ number of possible multiple s-at faults. Fault driven based test generation method is therefore not possible. On the other hand, random test generation cannot guarantee a high fault coverage and also is computationally costly.

[JACO 92] proposed an algorithm to modify an irredundant two-level multi-output PLA such that a test set for all single s-at faults is capable of detecting any multiple s-at faults. However, his work is limited to PLA applications.

[COX 88] proposed a fault diagnosis algorithm which was based on the A-16 alphabet to keep track of the set of unique line values possible in the presence of any single or multiple faults. However, the work was based on pseudo-random test generation. They do not have a solution to generate a test for multiple fault coverage. On the other hand, the algorithm may be used to show the effectiveness of a test generation method.

Similar work was carried out [KARK 94]. Fault dropping technique was used for fault analysis. Also, the work was based on pseudo-random test generation which have no guarantee that high fault coverage can be obtained.

[MACI 95][TAKA 91] proposed test generation algorithm for multiple fault coverage. However, simulation results showed that there were large variation of fault coverage between different benchmark circuits. It is believed that there is no single solution that can solve all problems.

[JONE 94][SETH 77] focused on multiple fault coverage problems in parity tree networks. Test generation algorithms were proposed to detect all single and multiple s-at faults.

[NANY 88] studied the multiple s-at fault testability of self-testing checkers. The simulation results showed that the fault coverage decreased as number of multiple faults increased for codeword testing in STCs. The use of non-codewords in off-line production testing to detect multiple s-at faults in the checkers was then proposed. However, in many systems, the checkers are not directly accessible from the primary inputs. Also, the functional blocks are not capable of generating non-codewords during normal operation. It means that additional hardware is required to provide the non-codewords to the checker.

More recently, [REDD 94] proposed a test generation algorithm for two-rail and parity checkers which is based on the output sequence analysis. Comparing this work with [JONE 94][SETH 77], there are many redundancies in the resultant test set. Therefore, the test set is not minimised.

Recall that SCCs are used in highly reliable and safe-critical applications. The TSC, SFS and SCD properties are defined based on single fault assumption. Under the occurrence of multiple faults, these properties would be invalidated. To discover the multiple faults in production stage becomes essential. Therefore, the idea of merging off-line testing approach for multiple faults with on-line testing capability becomes very attractive. The fundamental benefit of implementing a multiple fault detection algorithm is that it can provide good quality of the system during production test, resulting in higher mean-time-to-failure(MTTF). The effectiveness of on-line testing then becomes higher.

## 4.2 TEST GENERATION IN TWO-RAIL CHECKERS

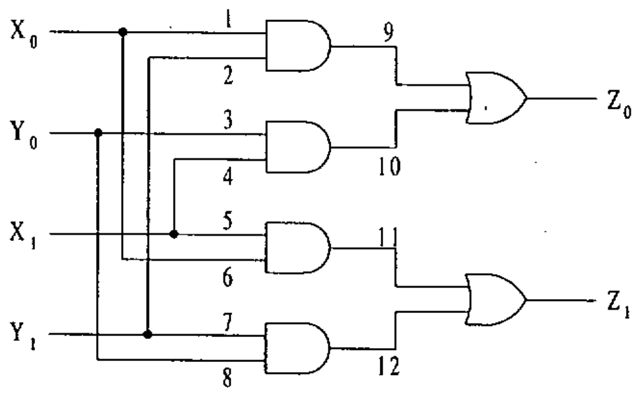
As discussed in previous section, multiple s-at fault detection algorithm is hard to develop. It is believed that there is no general solution to solve the multiple fault coverage problem. However, some circuit structures are highly testable for multiple faults such as two-level circuits, tree structured circuits, etc. In this research, we have studied the multiple fault detection algorithm of two-rail and parity checkers. Two-rail checkers(TRCs) are the most commonly used STCs. It is commonly used, for example, to multiplex outputs from different checkers to form a final checker. Furthermore, a pair of identical functional blocks and a two-rail checker(TRC) as a comparator can form the most simplest self-checking system. And parity checkers are usually used in bus-based system and memory modules.

A TRC compares two words that should normally be bit-by-bit complementary. If the two words are complementary, the checker will indicate a fault-free signal(01 or 10) at the two outputs, $Z_0$ and $Z_1$. And if the checker itself is faulty or the two words are not complementary, it will produce an error signal(00 or 11) at the two outputs, $Z_0$ and $Z_1$. Figure 4.1 shows three possible designs of a 2-pair TRC where $(X_i, Y_i)$ is an input pair. The circuits implement the following Boolean function:

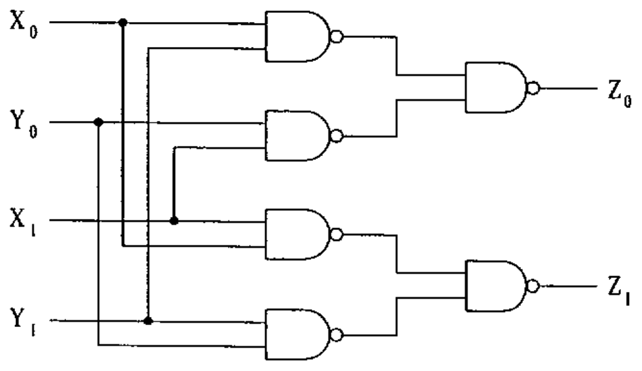
$$Z_0 = X_0 \bullet Y_1 + Y_0 \bullet X_1$$

$$Z1 = X_0 \bullet X_1 + Y_0 \bullet Y_1$$

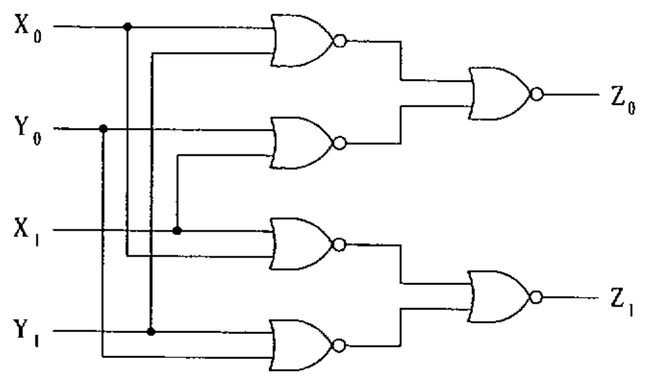
In general, an n-pair TRC can be implemented by cascading 2-pair TRCs as basic building blocks. Figure 4.2 shows two examples of TRC with a simplified TRC block diagram as basic building block. Each block is a 2-pair TRC which can be implemented by any one of the designs as shown in Figure 4.1. Each arrow-line represents a pair of input lines.



(a)



(b)



(c)

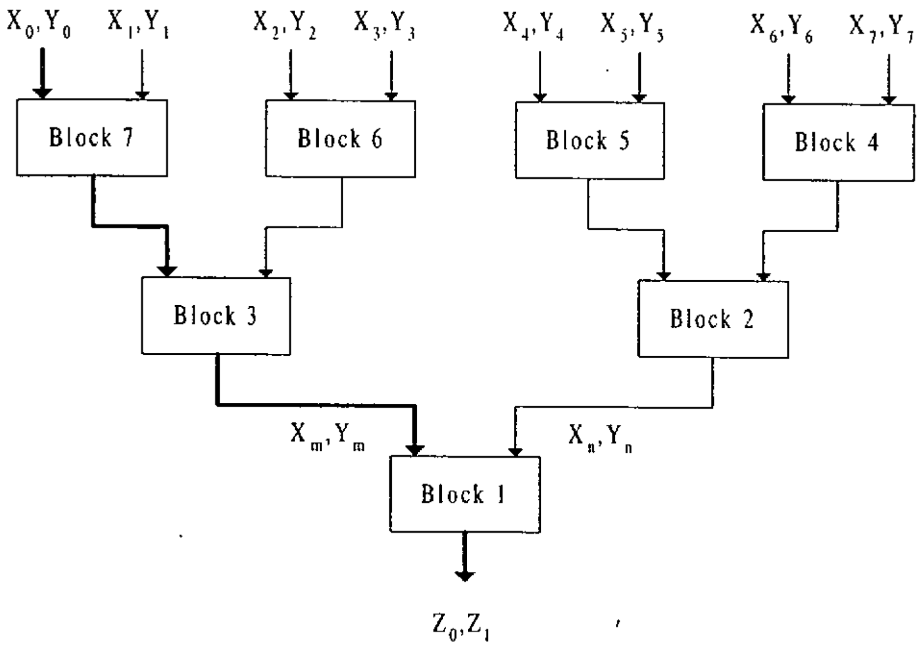
**Figure 4.1** Three types of 2-pair TRC, (a) AND-OR, (b) NAND, and (c) NOR gates implementation.

### 4.2.1 Test Generation Algorithm

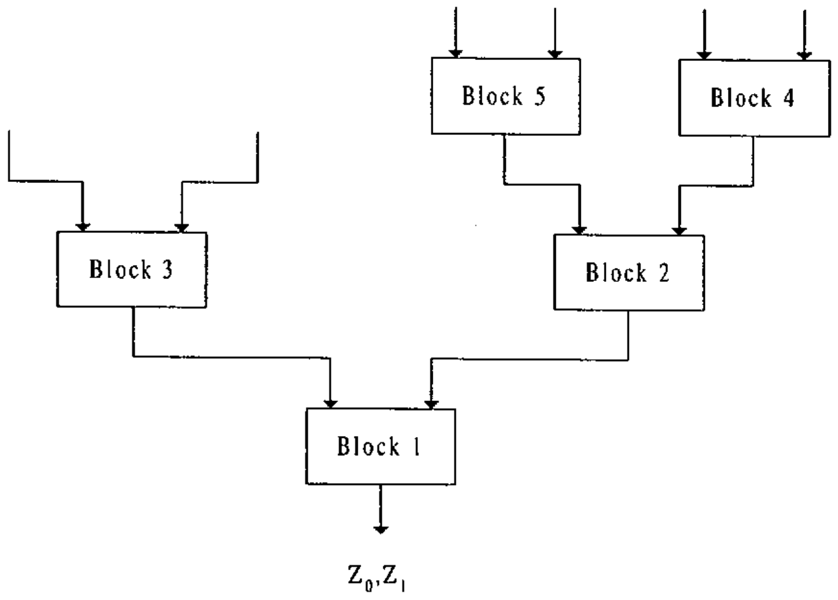
In our multiple s-at fault analysis, the circuit configuration is based on the circuit diagram as shown in Figure 4.1(a). Fault is detected based on an output sequence analysis approach which is similar to the one proposed in [REDD 94]. If the output sequence is not the same as the fault-free one, faults are detected. To simplify the analysis, we require that the output will toggle between two consecutive input vectors. So, if one of the following conditions are fulfilled, faults will be detected.

1) The output is a non-codeword,

2) The output is not toggled between two consecutive test vectors.



(a)



(b)

**Figure 4.2** Construction of TRCs using the 2-pair TRC as basic building blocks, (a) complete tree structured 8-pair TRC, and (b) incomplete tree structured 6-pair TRC.

## *A. Fault Model*

In self-testing checker, it has been found that all single s-at faults can be detected only by codeword inputs in [ADER 73][SMIT 78][NICO 88]. It is shown in [NANY 88] that a minimum 4-codeword test set is enough to detect all single s-at faults in a 2-pair TRC. But the fault coverage will be reduced as multiple faults occur. In this subsection, we shall define the fault model that our analysis is based on.

Consider the 2-pair TRC as shown in Figure 4.1(a). After fault collapsing, 12 unique faults are obtained in a 2-pair TRC. Lines 1 to 8 are modelled as s-at 1 faults and lines 9 to 12 as s-at 0 faults. Now, we can formally define our multiple fault model. Let Fs denote the set of s-at faults so that each block in the TRC will have a set Fs consisting of these 12 faults. The collection of Fs of all blocks in the checker constitutes a complete fault set F. The multiple faults we shall consider are the combination of all faults of all multiplicity in F.

## *B. Test Generation Procedure*

Now we present the test generation method and show the validity of the method. In this method, the test generation procedure works backward from primary output to primary input, on a block-by-block basis. The test vectors for a block are derived based on the analysis of the output sequence of the block. We introduce our method by the following example.

Refer to Figure 4.2(a). If the outputs ($Z_0$, $Z_1$) of block-1 are ({0101}, {1010}), the inputs of block-1 ($X_m$, $Y_m$, $X_n$, $Y_n$) derived are ({0011}, {1100}, {0110}, {1001}). Note that {} denotes a sequence of bits. It can easily be seen that all codewords are presented at the inputs of block-1. Based on this observation, a set of rules is derived for the test generation procedure:

1) Only one input pair is allowed to change between two consecutive test vectors. Although any odd number of input pair change will toggle the output, we change only one pair to simplify the test generation.

2) All possible codewords must be presented at the inputs of each block. Based on the example above( Figure 4.2(a)), block-2 will not have enough codewords at the inputs. Additional data outputs are therefore required. This is achieved by duplicating the original outputs sequences ({0101}, {1010}) to ({0101 0101}, {1010 1010}). The inputs of block-1 then become ({0011 0011}, {1100 1100}, {0110 0110}, {1001 1001}) as shown in Figure 4.2(a). It is easy to show that the inputs of block-2 are ({0001 1110}, {1110 0001}, {0111 1000}, {1000 0111}). All codewords are presented at the inputs of block-2. Similar procedures can be performed for other blocks.

3) The test generation procedure described in Rule 2) is applied iteratively level-by-level until the primary inputs are reached.

Based on the proposed test generation procedure, a complete test set for the X-terminal inputs(denoted from left-to-right) of the 8-pair TRC of Figure 4.2(a) is shown in Table 4.1. Note that only the test vectors of the X-terminal inputs are shown. The Y-terminal inputs are only the complements of the corresponding X-terminal inputs. With the test generation procedure demonstrated above, the importance of "ease of use" cannot be over-emphasised.

**Table 4.1** Test vectors determined for Figure 4.2(a).

| X-Inputs | bit sequence |
|---|---|
| $X_7$ | 0111 1111 1000 0000 |
| $X_6$ | 0000 0111 1111 1000 |
| $X_5$ | 0001 1111 1110 0000 |
| $X_4$ | 0000 0001 1111 1110 |
| $X_3$ | 0011 1111 1100 0000 |
| $X_2$ | 0000 0011 1111 1100 |
| $X_1$ | 0000 1111 1111 0000 |
| $X_0$ | 0000 0000 1111 1111 |

*C. Test Length Comparison*

The number of test vectors required by our method is determined by the number of input pairs, k, and the number of levels, d. For a tree structured network with highest level d, if there are no primary inputs attached to blocks with more than 1 level difference, we have

$$d = \lceil \log_2 k \rceil$$

For a single level tree network, the number of input vectors required is 4. With each additional level of blocks, one duplication of output sequence is required. So the test length, L, generated by our method is,

$$L = 2^{d+1}$$

Figure 4.3 shows the test length comparison between our method to the one given by [REDD 94].

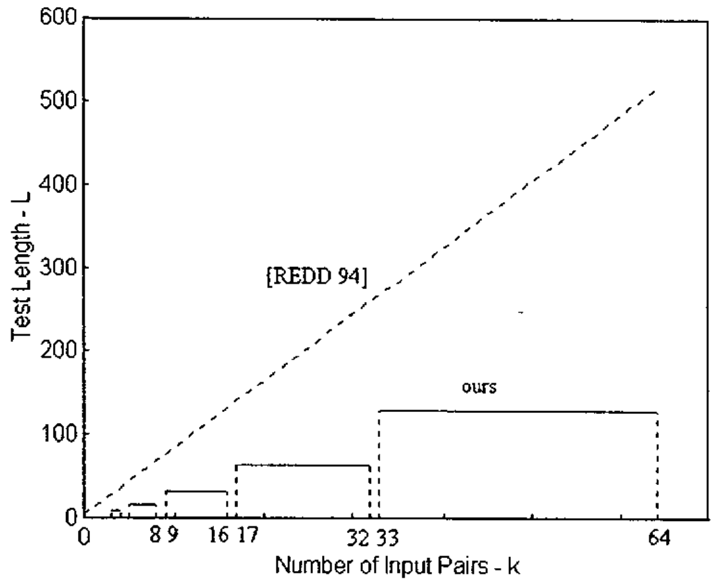


**Figure 4.3** Test length comparison in terms of input pairs.

### 4.2.2 Fault Simulation

Fault simulation was done by C programming. A 16-pair TRC, which contains totally 180 number of unique faults after fault collapsing, was used to verify the test generation method. We determined the multiple fault coverage by exhaustively injecting all combination of the unique faults up to 4 multiplicity and randomly generating 1 million multiple faults all any multiplicity in the circuit. Then, the determined test set is applied to the circuit for fault coverage evaluation. Results showed that our test set is sufficient to detect the injected multiple faults and all determined test vectors are necessary in the fault detection process.

### 4.2.3 Validity of the Test Generation Method

Our test generation method is based on the assumption that the output function of TRC will not complement for multiple faults. We prove this by considering the fault effect on each block in a tree structured checker such as the one shown in Figure 4.2(a).

*Lemma*: There are no multiple faults such that the output responses are the same as fault-free ones or their complements in our test.

*Proof*: To detect multiple faults in a 2-pair TRC, four codewords are enough. We conduct the proof procedure in two steps. In step 1, a detailed multiple fault analysis of the 2-pair TRC is carried out based on the fault diagnosis scheme described in [COX 88] and [KARK 94]. The results of this analysis demonstrate the validity of the four codeword test in the presence of any multiple faults of all multiplicity. In step 2, we extend the multiple fault analysis to the k-pair TRC.

Table 4.2 summarises the results after step 1, where a set of input codewords ($T_1 = 0101$, $T_2 = 1001$, $T_3 = 1010$, $T_4 = 0110$) is applied pair-wise consecutively to the 2-pair TRC as shown in Figure 4.1(a). Each row shows all the possible logic values of a particular line in the presence of any number of s-at faults in the checker. Any line values that invalidate the expected output line values are marked by a cross-over bar. In this case, the expected output line values are the same as fault-free ones, which are $Z_0$ = {0101} and $Z_1$ = {1010}.

**Table 4.2** Multiple fault diagnosis of a 2-pair TRC.

| Line label | Test Pair { $T_1, T_2$ } | Test Pair { $T_2, T_3$ } | Test Pair { $T_3, T_4$ } |
|---|---|---|---|
| 1 | { 01 / ~~11~~ } | { 11 / 11} | { 10 / 11 } |
| 2 | { 11 / 11 } | { 10 / ~~11~~} | { 00 / ~~11~~ } |
| 3 | { 10 / 11 } | { 00 / ~~11~~ } | { 01 / ~~11~~ } |
| 4 | { 00 / ~~11~~ } | { 01 / 11 } | { 11 / 11 } |
| 5 | { 00 / ~~11~~ } | { 01 / 11} | { 11 / 11 } |
| 6 | { 01 / 11 } | { 11 / 11 } | { 10 / ~~11~~ } |
| 7 | { 11 / 11 } | { 10 / 11 } | { 00 / ~~11~~ } |
| 8 | { 10 / ~~11~~ } | { 00 / ~~11~~ } | { 01 / 11 } |
| 9 | { 01 / ~~00~~, ~~11~~ } | { 10 / ~~00~~, ~~11~~ } | { 00 / ~~10~~, 00, ~~11~~ } |
| 10 | { 00 / ~~10~~, 00, ~~11~~ } | { 00 / ~~01~~, 00, ~~11~~ } | { 01 / ~~00~~, ~~11~~ } |
| 11 | { 00 / ~~01~~, 00, ~~11~~ } | { 01 / ~~00~~, ~~11~~ } | { 10 / ~~00~~, ~~11~~ } |
| 12 | { 10 / ~~00~~, ~~11~~ } | { 00 / ~~10~~, 00, ~~11~~ } | { 00 / ~~01~~, 00, ~~11~~ } |
| $Z_0$ | { 01 / ~~10~~, ~~00~~, ~~11~~ } | { 10 / ~~01~~, ~~00~~, ~~11~~ } | { 01 / ~~10~~, ~~00~~, ~~11~~ } |
| $Z_1$ | { 10 / ~~01~~, ~~00~~, ~~11~~ } | { 01 / ~~10~~, ~~00~~, ~~11~~ } | { 10 / ~~01~~, ~~00~~, ~~11~~ } |

- Line labels are referred to Figure 4.1(a)
- Line values are denoted as { fault-free values / all possible faulty values }

From the results obtained, it can be seen that no multiple faults can cause the two outputs of a 2-pair TRC to have the fault-free values.

**Table 4.3** Diagnosis of two-rail checker having the outputs responses equal to the complement of fault-free ones.

| Line label | Test Pair { $T_1$, $T_2$ } | Test Pair { $T_2$, $T_3$ } | Test Pair { $T_3$, $T_4$ } |
|---|---|---|---|
| 1 | { 01 / 11 } | { 11 / 11} | { 10 / 11 } |
| 2 | { 11 / 11 } | { 10 / 11} | { ~~00~~ / 11 } |
| 3 | { 10 / ~~11~~ } | { ~~00~~ / 11 } | { 01 / 11 } |
| 4 | { ~~00~~ / 11 } | { 01 / ~~11~~ } | { 11 / 11} |
| 5 | { ~~00~~ / 11 } | { 01 / 11} | { 11 / 11 } |
| 6 | { 01 / ~~11~~ } | { 11 / 11 } | { 10 / 11 } |
| 7 | { 11 / 11} | { 10 / ~~11~~ } | { ~~00~~ / 11 } |
| 8 | { 10 / ~~11~~ } | { ~~00~~ / 11 } | { 01 / 11 } |
| 9 | { ~~01~~ / 00, ~~11~~ } | { ~~10~~ / 00, ~~11~~ } | { ~~00~~ / 10, ~~00~~, ~~11~~ } |
| 10 | { ~~00~~ / 10, ~~00~~, ~~11~~ } | { ~~00~~ / 01, ~~00~~, ~~11~~ } | { ~~01~~ / 00, ~~11~~ } |
| 11 | { ~~00~~ / 01, ~~00~~, ~~11~~ } | { ~~01~~ / 00, ~~11~~ } | { ~~10~~ / 00, ~~11~~ } |
| 12 | { ~~10~~ / 00, ~~11~~ } | { ~~00~~ / 10, ~~00~~, ~~11~~ } | { ~~00~~ / 01, ~~00~~, ~~11~~ } |
| $Z_0$ | { ~~01~~ / 10, ~~00~~, ~~11~~ } | { ~~10~~ / 01, ~~00~~, ~~11~~ } | { ~~01~~ / 10, ~~00~~, ~~11~~ } |
| $Z_1$ | { ~~10~~ / 01, ~~00~~, ~~11~~ } | { ~~01~~ / 10, ~~00~~, ~~11~~ } | { ~~10~~ / 01, ~~00~~, ~~11~~ } |

- Line labels are referred to Figure 4.1(a)
- Line values are denoted as { fault-free values / all possible faulty values }

Next, we consider the multiple fault detection problem in a k-pair tree-structured TRC. Refer to Table 4.3. There is only one input pair that can be changed between two consecutive input test patterns. Consider the high-lighted path from primary inputs to primary outputs in Figure 4.2(a). Suppose that block $2^d$-1 is the first faulty block along the path. After applying a two-pattern test, there are three possible fault effects occurring at the output of this faulty block: 1) one output line carries unidirectional transition(i.e. either {01} or {10}) while the other output line remains unchanged or both lines carry unidirectional transition, 2) both output lines stay unchanged(i.e. {00} or {11}), 3) the outputs are complement to the fault-free ones. For a fault going undetected, there must be other faulty block(s) along the path(e.g. blocks 3 and 1) such that these faulty outputs will be converted to fault-free ones. In the following, we shall prove that for all of the above three fault effects, our test method is capable of detecting the existence of multiple faults.

1) One output line carries unidirectional transition while the other output line remains unchanged or both lines carry unidirectional transition. It can be shown that there are no faults such that a faulty 2-pair TRC can produce fault-free outputs. The reason for this is that a 2-pair TRC is an unate function which is not capable to generate any EXOR operation when it is faulty. Similarly, no fault-free outputs can occur when both lines carry unidirectional transition.

2) Both output lines stay unchanged. The outputs of the faulty 2-pair TRC under multiple faults may appear as codewords. However, the output would not be toggled in the two-pattern test. This fault effect will propagate to the primary outputs and be eventually detected.

3) The outputs are complement to the fault-free ones. In Table 4.2, it has already been proved that no multiple faults can result in fault-free outputs. Therefore here we only need to show that no multiple faults can occur such that the outputs are the complements of the fault-free ones. We show this by referring to Table 4.3, which is generated based on the assumption that the outputs are complement of the fault-free ones. Consider as an example test ($T_1$, $T_2$). In order to produce ({10}, {01}) at the outputs, line 9 must be {00} which is not generated by the AND gate. So line 9 can be assumed to be s-at 0. But in test ($T_3$, $T_4$), line 9 must carry values {10} such that outputs can be complement to the fault-free ones. So there is a contradiction between tests ($T_1$, $T_2$) and ($T_3$, $T_4$). Therefore, it is impossible for a faulty TRC block to generate any faulty outputs which are complemented to the fault-free ones for our test.

Through such detailed analysis, it can be proved that there are no faults such that the outputs are equal to the fault-free ones, or their complements. On the other hand, the erroneous outputs(under the occurrence of either fault effect 1 or fault effect 2) generated by a faulty block along the path can propagate to the primary output lines. All multiple faults can thus be detected by our test.

### 4.2.4 Implementation of the Test Generation Method

In order to make use of the test generation method more effectively, a two-step method to implement the test generation algorithm was proposed In the first step, every pair of input lines of a TRC is labelled by an unique number according to a input pair number assignment method. So that, the corresponding bit sequence of every line through the test can be easily determined. If the line is primary input, test set is derived. In the second step, a novel analytic scheme is used to analyse the test set, to enable the user to simplify the implementation of the test generation procedure.

## A. *Input Pair Number Assignment Method*

For the purpose of identification, each input pair of a block is labelled with a number known as input pair number. The method of assignment input pair numbers is worked level-by-level. The level of a block is defined as the distance of the block from the output block. For example the block nearest to the primary outputs is known as level-1 block. Referring to Figure 4.2(a), block-1 is known as level-1 block, and blocks 4 to 7 are known as level-3 blocks. For level-1 inputs(block-1 inputs), the assignment of the input pair numbers is arbitrary. For other upper level inputs, we apply the following rules to determine the input pair numbers:

1) The number of one input pair of a block is assigned the same number as the output pair of the same block.

2) The number of the other input pair is determined by the formula $N(i)+2^{m-1}$, where N(i) is the number of the first input pair of block-i that has already been defined in Rule 1), and m is the level of block-i.

Figure 4.4 further illustrates the input number assignment method of a 8-pair TRC. A set of notations are defined as follows:

- The input sequence of input pair n of a block at level-m is denoted as ($X_{m,n}$, $Y_{m,n}$).
- The output sequence of block-i is ($Z0_i$, $Z1_i$).

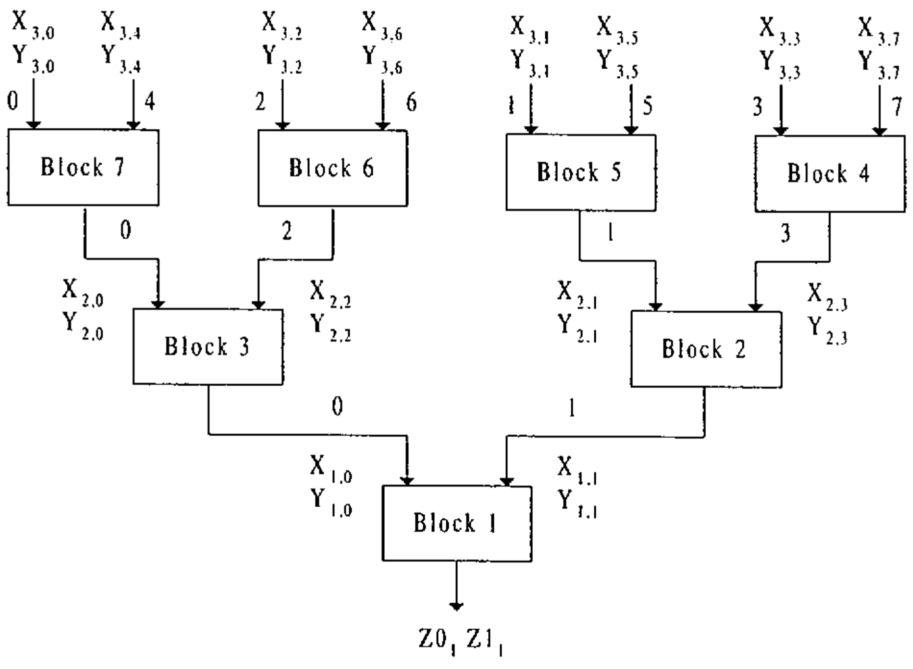


**Figure 4.4** An example of 6-pair TRC.

After introducing the input pair number assignment method, we now formulate our method in a formal way. Firstly, we need to assign each input pair of each block at each level with an unique number as shown in Figure 4.4. Again, we only show the test sequence generation for the X-terminal inputs because the Y-terminal inputs are only complements of the corresponding X-terminal inputs.

Let $S_m$ be the *basic input sequence* of level-m block and $S_0$ be the primary output sequence. Based on the assumption of fault-free output, $S_0$ will be an alternating sequence of the form {01010101010101......}. Here we introduce a 2-step process using the following two equations to determine the required inputs of each block:

$$S_m(j) = S_0(\lfloor j/2^m \rfloor) \tag{1}$$

In the first step, Equation (1) is used to determine the bit pattern of the basic input sequence at level-m, where j represents the position of individual bits. Here the bits are numbered 0,1,2,3,... from left to right in a sequence.

$$X_{m,n}(j) = S_m(j+n) \tag{2}$$

In the second step, Equation (2) is used to determine the exact bit sequence of input pair n at level-m.

As an example, consider the test vector at the input terminal $X_{3,4}$ of an 8-bit TRC. First, the basic input sequence of level-3 blocks is determined:

$$S_3(j) = S_0(\lfloor j/8 \rfloor)$$

$$S_3 = \{0000000011111111.......0000000011111111........\}$$

By substituting input pair number 4 and level-3 into equation (2)

$$X_{3,4}(j) = S_3(j+4)$$

$$X_{3,4} = \{000011111111000000001111....0000000011111111....\}$$

This example demonstrates that the proposed method is very simple and straight forward to implement.

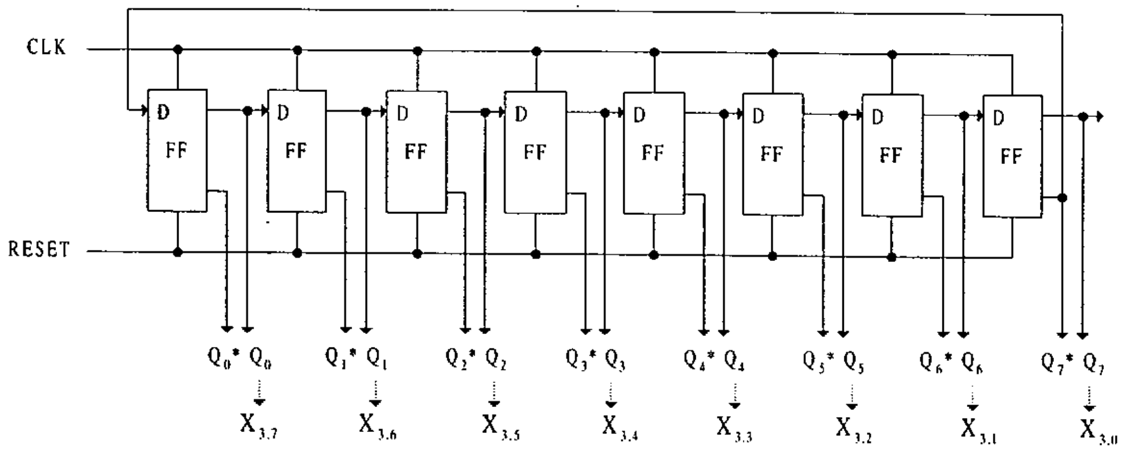
*B. Implementation Using CSR*

In a complex digital system, it is always hard to access individual blocks from primary inputs. Cyclic shift register(CSR) based on built-in self-testing(BIST) approach [ABRA 90][FUJI 85] was proposed to generate the required test vectors directly at the input of the circuit-under-test. We now show that out test can be easily implemented by CSR. By-rearranging the test vectors developed for the TRC as shown in Figure 4.2(a), Table 4.1 becomes:

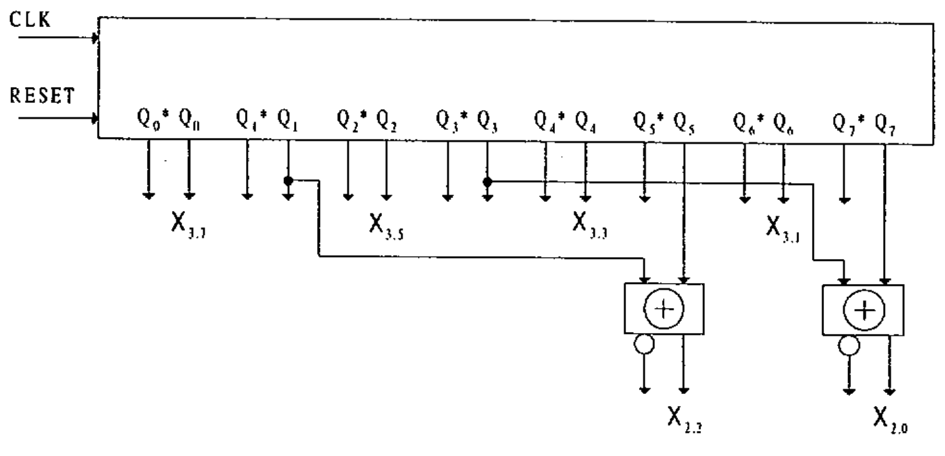
**Table 4.4** Re-arrangement of Table 4.1.

| X-Input | bit sequence |
|---|---|
| $X_{3,7}$ | 0111 1111 1000 0000 |
| $X_{3,6}$ | 0011 1111 1100 0000 |
| $X_{3,5}$ | 0001 1111 1110 0000 |
| $X_{3,4}$ | 0000 1111 1111 0000 |
| $X_{3,3}$ | 0000 0111 1111 1000 |
| $X_{3,2}$ | 0000 0011 1111 1100 |
| $X_{3,1}$ | 0000 0001 1111 1110 |
| $X_{3,0}$ | 0000 0000 1111 1111 |

It is clear that these sequences (as denoted in Table 4.4) can be generated by a simple CSR as shown in Figure 4.5(a). The 8-pair TRC checker requires only 8 shift register cells to generate the required set.



(a)



(b)

**Figure 4.5** CSRs for (a) Figure 4.2(a), and (b) Figure 4.2(b).

Note that although the example given in Figure 4.5(a) is for a complete tree structured TRC, the same approach can be applied to any incomplete ones. Figure 4.5(b) shows how easy it is to modify the structure in Figure 4.5(a) in order to generate test vectors required for an incomplete tree such as the one shown in Figure 4.2(b). Detailed labelling can be referred to Figure 4.4.

### 4.2.5 Application of Test Generation in Parity Checkers

In previous section, a simple test generation method for two-rail checker is presented. Here we show how the test developed in Section 4.2.1 is applicable to the parity checker. A parity checker consists of two parity trees $T_1$ and $T_2$. Design methods for self-testing embedded parity checkers based on XOR gates have been reported in [KAHA 84]. Because the two parity trees are totally separable from each other, we will explain our test generation method based on single tree only.

#### *A. Fault Model of Parity Tree*

Consider the different parity tree configurations shown in Figure 4.6, the two parity trees requires same number of gates but Figure 4.6(a) gives a less gate delay. In this paper, our simulation is based on parity tree with configuration such as the one shown in Figure 4.6(a). Indeed, our method can be applied to Figure 4.6(b) configuration too. The fault model considered is all combination of stuck-at faults in the parity tree.

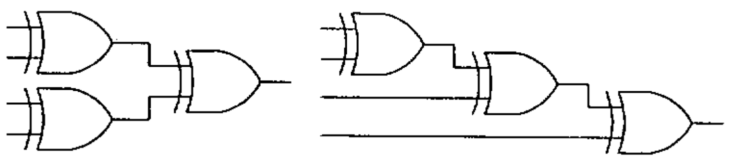


**Figure 4.6** Examples of two different parity tree configurations.

*B. Test Generation Procedure*

In this section, the test generation method developed in Section 4.3.1 will be applied to the parity checker. Also the effectiveness of this method in XOR parity tree will be shown. Following the procedures in Section 4.3.1, the output is set to alternating 0 and 1 as shown in Figure 4.7. After applying rules 1, 2 and 3, similar results with those shown in Figure 4.7 are obtained.

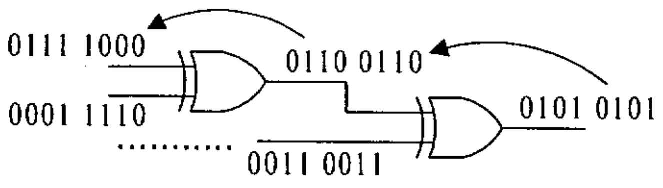


**Figure 4.7** Example of test generation of parity tree.

*C. Validity of the Test Generation Method for Parity Checker*

The validity of the test in TRC is provided in Section 4.2.3. Since the same procedure developed for TRC is applied to the parity tree, we only need to prove the validity of the procedure in parity checker. To prove the validity of the method, we have the follow lemma:

*Lemma 2: There are no multiple faults in the XOR-type parity checker such that the output response is same as or complement of the fault-free one.*

*Proof: Based on Figure 4.7, after completing the test generation, there is no bit sequences in any input of a gate which is equal to or complement of the fault-free output sequences. When one of the input of an XOR gate is stuck-at 0 or 1, the output bit stream of the gate will be same as or complement of the input respectively. The fault-free signal of any output of XOR gate can be only generated by the particular input signal. So the only way to generate the alternating 0 and 1 at the output of the tree is when it is fault-free.*

*D. Implementation of Test Generation Method*

In Section 4.2.4, a simple CSR is provided to implement the test generation method. Based on the test generation on an 8-input parity tree, same set of test vectors are obtained as shown in Section 4.2.4. Without any major modification, same CSR in Figure 4.5(a) can be used with Q terminal outputs or using the complement of Q terminal outputs. Without proof, anyone of the two sets of outputs of CSR (Q or Q*) can be used to generate the required test vectors to achieve the multiple faults coverage. This observation gives us an advantage of using same CSR for both parity checker and TRC in the same circuit such that overall additional hardware is reduced. In general, this application is not limited for complete binary tree but can also be applied to incomplete one.

## 4.3 SUMMARY

In this section, we have introduced a test generation method which requires codewords only to detect all multiple stuck-at faults in two-rail and parity checkers. The main objective of developing the test generation method for multiple fault coverage is to provide a high quality test in the production testing phase, such that the system can be shown to be of higher quality, resulting in longer mean-time-to-failure. Then, the implementation of self-checking technique is more effective. However, there are limitations in test set development. The first one is that the checker is not directly accessible. The second one is that without extra hardware only codewords are presented at the input of the checker during normal operation. We have overcome these two problems by using codewords testing with a simple output sequence analysis. Furthermore, our method is very suitable for implementation using BIST approach. In summary, we think that currently there is no one test generation algorithm to detect all multiple faults in an efficient manner. For some circuit types, there is no guarantee that multiple fault is detectable. The test generation algorithm becomes application oriented. However, it is found that tree structured networks are highly testable for multiple faults.

# Chapter 5

# Testability Analysis and Detection of Transistor Faults in CMOS VLSI Circuits

## 5.1 INTRODUCTION

In this chapter, we consider the testability of static and domino-CMOS logic circuits under open and bridging faults. For many years, classical stuck-at fault model has been widely adopted because it is very easy to use and it covers large number of physical defects. As the CMOS process becomes the dominant technology in VLSI design, the effectiveness of stuck-at fault in CMOS VLSI circuit should be reviewed. Open and bridging faults are two main fault types in VLSI circuit. However, the possible number of these two types of faults are very large. In this chapter, we restrict the fault model to single fault only.

As shown in [GALI 80][LEVI 81][MALA 82][MILL 91][SHEN 88], the requirement of detecting those faults are more complicated than logical fault, most ST checkers are based on the logical fault model at the gate-level [BUSA 94][JOHN 89][KHAK 84][LALA 85]. Unfortunately, logical fault model such as stuck-at fault is insufficient and inadequate to represent some physical defects in CMOS VLSI circuits. Large number of faults cannot be explained by a logical fault model. [MILL 91] has reviewed the behaviours of self-testing(ST) CMOS checkers under transistor-level faults such as bridging, transition and stuck-open faults. [NICO 91] has studied extensively the effect of bridging faults in several ST checkers based on different implementation technologies.

## 5.2 TESTABILITY ANALYSIS OF STATIC CMOS LOGIC CIRCUITS

In this section, the testability of static CMOS logic circuits under the occurrence of bridging faults and stuck-open faults will be discussed.

### 5.2.1 Stuck-open Fault

Open fault is a fault caused by a break at a conductor. In the following discussion, an open fault may be considered as a break at the gate node of a transistor which is named as transistor stuck-open fault. As discussed in Chapter 2, detection of stuck-open fault always requires two-pattern test. The first test vector initialises the output of a gate to a known state. Then, another test vector, which normally should result in complement output of the previous test vector, will be applied. If the conducting path contains a stuck-open fault, then the output of the gate will enter high impedance state and will be same as the previous one. So, the fault is detected. Therefore, a combinational circuit can behave as a sequential one. However, as studied by [MILL 91], an output hazard can invalidate the test. If inputs of a gate do not change simultaneously, the output may be charged(discharged) during transition. As a result, the circuit behaves correctly for the two-pattern test. The fault will be masked. However, if the transition is limited, the output hazard can be eliminated. On the other hand, some gates such as XOR gates requires complementary input signals, hazard may always exist in these gates. So, the use of this type of gates should be kept to the minimum so that the probability of an output hazard occurring can be reduced.

Stuck-open fault can result in sequential behaviour, but still produce logical error. Specific sequence of test vectors is required to detect this fault. Despite of the output hazard during transition by limiting the transition between test vectors, [MILL 91] showed the probability of detecting stuck-open faults with randomly generated test vectors for two-rail and parity checkers for a probabilistic analysis of stuck-open fault detection. Given the confidence level $C_L$, which is the probability that the fault will be detected by L random test vectors, if L is comparable to the number of possible input codewords, then the checker will remain ST for the stuck-open fault. The confidence level and test length L is related by Equation 5.1.

$$C_L = \sum_{m=1}^{L} (-1)^{m+1} \binom{L-m}{m} d^m \qquad (5.1)$$

where d is the probability of randomly applying two vectors to detect the fault.

Therefore, the values d for the checkers can be obtained. However, when d is small, the probability of detecting the corresponding fault is low, it requires to verify the CD property of the checker under the occurrence of the fault.

### 5.2.2 Bridging Fault

Fault which two lines are connected together is known as bridging fault. The effect of the bridging faults is dependent on the conductance of pull-up network, bridging resistance and the conductance of pull-down network. To model the bridging fault is ineffective because the resistance of the bridging can be any values and the occurrence of bridging faults is geometry dependence. Consider that if only intratransistor bridging faults shall occur, the total number of bridging faults will be 4 times of the number of transistors. Large number of possible fault can occur such that it is hard to handle each fault individually. On the other hand, the bridging resistance can be any values between perfectly short to open circuit. The effect of the fault decreases gradually as the resistance increases. However, logic testing is not effective to detect the bridging faults. Let us consider the following example:

**Example 5.1.** Consider that a bridging between the two outputs of the inverters as shown in Figure 5.1.

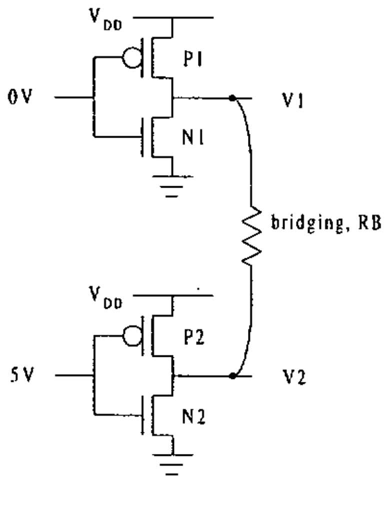


**Figure 5.1** Bridging between two outputs of inverters.

The transfer characteristic of a NMOS transistor can be categorised into three regions: cut-off, linear and saturation regions [WEST 93]. The three regions can be described by the following equations.

Region 1. $I_{ds} = 0$, when $V_{in} < V_t$ cut-off region

Region 2. $I_{ds} = \beta((V_{gs} - V_t) - \frac{V_{ds}}{2})V_{ds}$, when $0 < V_{ds} < V_{gs} - V_t$ linear region

Region 3. $I_{ds} = \beta\frac{(V_{gs} - V_t)^2}{2}$, when $0 < V_{gs} - V_t < V_{ds}$ saturation region

where $\beta = \frac{\mu\varepsilon}{tox}\left(\frac{W}{L}\right)$

and $V_t$ is threshold voltage of the MOS device

$\mu$ is mobility of the majority carrier

$\varepsilon$ is permittivity of the gate insulator

W is channel width of the device

L is channel length of the device.

Assume that the leakage currents of transistor N1 and P2 are negligible, so that they can be ignored in the analysis. Generally, the operation of transistors P1 and N2 can be categorised into three conditions as denoted in Table 5.1. The first two conditions indicate the bridging fault results. The faulty outputs are logical and they can be detected easily by logic testing. However, in the third condition, non-logical error will be produced at the faulty output lines since both transistors are in linear region. So, we will have the following three equations:

$$I_{ds} = \frac{V1 - V2}{RB} \qquad (1)$$

$$I_{ds} = \beta_n((V_{DD} - V_{tn})V2 - \frac{V2^2}{2}) \qquad (2)$$

$$I_{ds} = -\beta_p((-V_{DD} - V_{tp})(V1 - V_{DD}) - \frac{(V1 - V_{DD})^2}{2}) \qquad (3)$$

**Table 5.1** The operation conditions of transistors P1 and N2 under bridging fault condition.

| Condition | P1 | N2 |
|---|---|---|
| 1 | Linear | Saturation |
| 2 | Saturation | Linear |
| 3 | Linear | Linear |

[LEE 96a][LEE 96b] solved these equations by iteration with common transistor gain factors and threshold voltages. Figure 5.2 shows the plotting of voltage outputs versus different bridging resistance RB. It can be noticed that the output can fall in any value between $V_{DD}$ and $V_{SS}$. Because a CMOS logic gate is a high gain amplifier, the degradation in input voltage may not affect the output voltage, but the timing may be affected. Hence, bridging fault is hard to be detected by logic testing. However, if we monitor the Ids of the circuit, we find that this current will be much larger than the normal value as illustrated in Figure 5.3. Based on this observation, various supply current testing techniques(or called Iddq testing) and test pattern generation methods for Iddq testing were developed [FERG 91][LEVI 81][MALA 82][MALY 92][MIUR 92][RAJS 95][STOR 91][TANG 95][WEN 97].

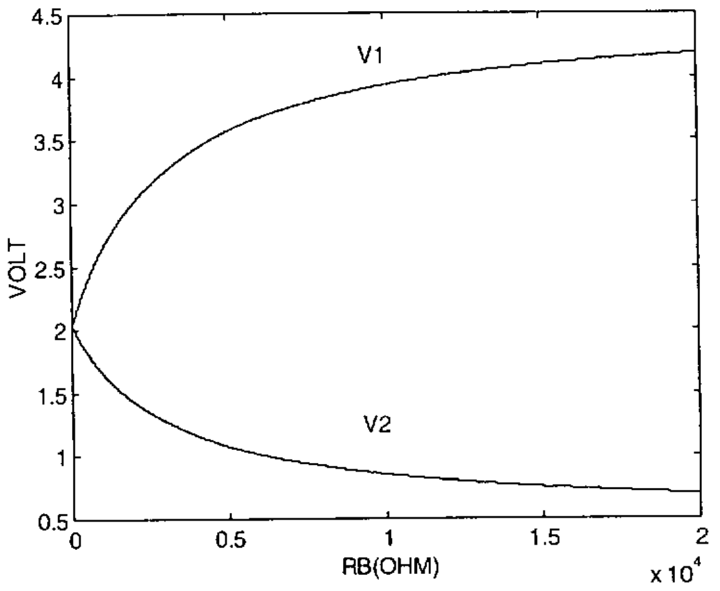


**Figure 5.2** Plotting of output voltages of two inverters with different bridging resistance.

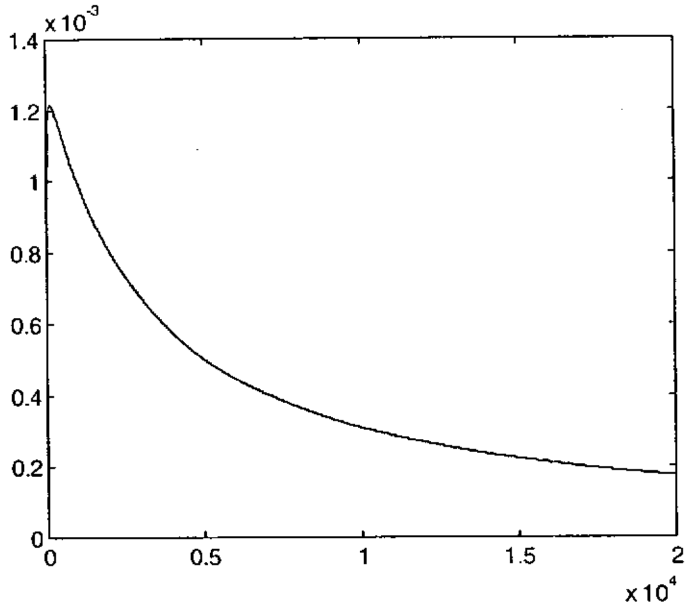


**Figure 5.3** Plotting of Ids versus different bridging resistance.

Iddq(supply current quiescent) testing bases on the fact that when a static CMOS logic gate is not switching, it draws no supply current except a very small leakage current. When a bridging fault occurs, a conducting path between $V_{DD}$ to ground node will be created to carry a large current as denoted in Figure 5.3. A sensing device is commonly used to monitor the supply current. When a fault is activated at the faulty site, it will be detected and indicated by the sensing device. The main advantage of Iddq testing technique is that the faulty output value does not need to propagate to the primary output. The test generation will become easier. However, Iddq testing is relative low in detection speed comparing to the logic testing. To speed up the testing process, a current monitoring device called built-in current sensor(BICS) is required to monitor the supply current [RAJS 95]. For high quality testing, the current resolution of BICS must be high. Furthermore, the device is inserted in the supply or ground line permanently, the normal operation speed of circuit under test(CUT) will be reduced. In summary, for a high quality system, Iddq testing technique should be implemented to provide a high quality test. On the other hand, trade-off between the quality of test and the speed of test must be considered. In the next section, we shall demonstrate the effectiveness of implementing Iddq testing into totally self-checking(TSC) system such that a high quality system can be achieved.

### 5.2.3 Integration of Iddq testing with TSC system

In this section, we propose a method to integrate a TSC system with Iddq testing approach. Later, we shall use an example of Iddq testing in a two-rail checker to demonstrate the effectiveness of the implementation and the use of test generation method proposed in Chapter 4 in Iddq testing.

*A. Iddq Testing Circuit*

Many kinds of built-in current sensor(BICS) were designed and implemented in the past [MIUR 92][MOZU 96]. In this study, we have designed a simple current sensor circuit that can be inserted in power line or ground line of the CUT as shown in Figure 5.4 to measure the supply current in the simulation. In [MOZU 96], a comparison of Idd and Iss BICS was studied. It is shown that the Iss BICS is faster and requires smaller area than Idd BICS. An Iss BICS also results in less degradation of performance in the CUT.

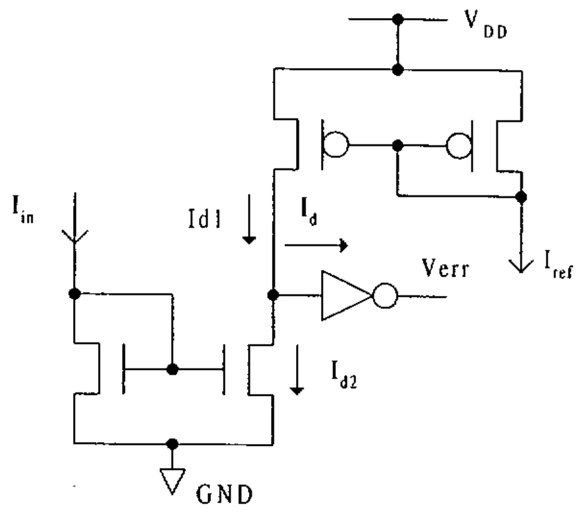


**Figure 5.4** Built-in current sensor.

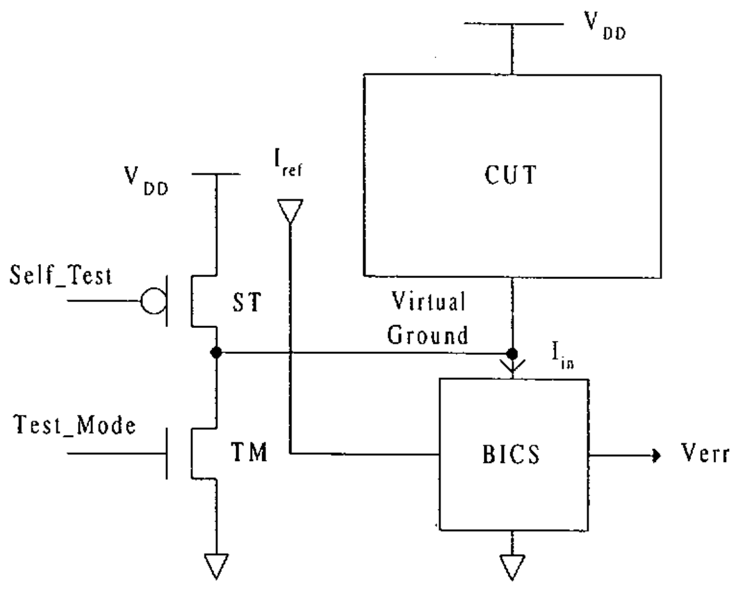


**Figure 5.5** Application of BICS.

As shown in Figure 5.5, a BICS is inserted between the ground node of the CUT and GND. Quiescent current will flow from $V_{DD}$ through the BICS to GND when suitable test vectors are applied to activate the bridging faults. Two extra transistors are added for self-testing and test enable of the BICS. Transistor ST allows high current flowing into the BICS so that the testing of the function of the BICS can be carried out independent of the CUT. Transistor TM is designed to have higher conductance than the input resistance of the BICS in order to bypass the current flowing into the BICS during normal operation mode.

The operation of the BICS as shown in Figure 5.4 is quite simple. Current $I_{d1}$ is mirrored from $I_{ref}$ through the PMOS current mirror. The quiescent current Iin from the CUT is mirrored to $I_{d2}$. Since, an inverter is a high gain amplifier, small variation in input voltage will result in very large output changing. If $I_{d1}$ is larger than $I_{d2}$, $I_d$ will be large enough to charge up the input of the inverter. The error flag will indicate fault-free logic 0 at the output. If $I_{d2}$ is larger than $I_{d1}$, the output of inverter will change to logic 1, indicating an error. In the study, we assume that sufficient attention is given to controlling the background leakage current. For the testing approach to remain effective, we must ensure that the magnitude of the fault-free quiescent current has to be estimated carefully. Furthermore, reference current $I_{ref}$ has to be selected properly to distinguish the faulty quiescent current state from the fault-free one. Figure 5.6 shows the operation of the BICS corresponding to a faulty circuit.

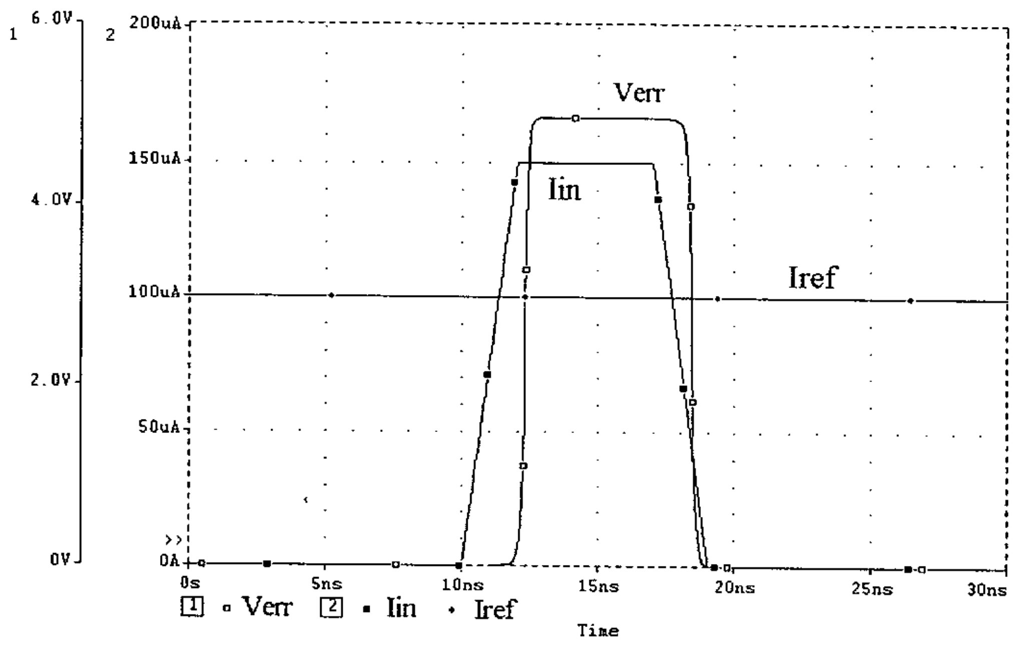


**Figure 5.6** Output of the BICS corresponding to a faulty circuit.

A TSC system by definition produces a two-bit output to indicate the error. A simple interface circuit is used to integrate the BICS output into the overall TSC fault indication circuit. It consists of a 2-input NOR gate and a 2-pair two-rail checker. The overall circuitry is shown in Figure 5.7. The NOR gate is used to translate the BICS output error signal E, into 2-output $C_0$ and $C_1$. An additional test control pin Test* is provided to mask or unmask the output of the BICS. Table 5.2 summaries the operation of the Iddq testing circuit.

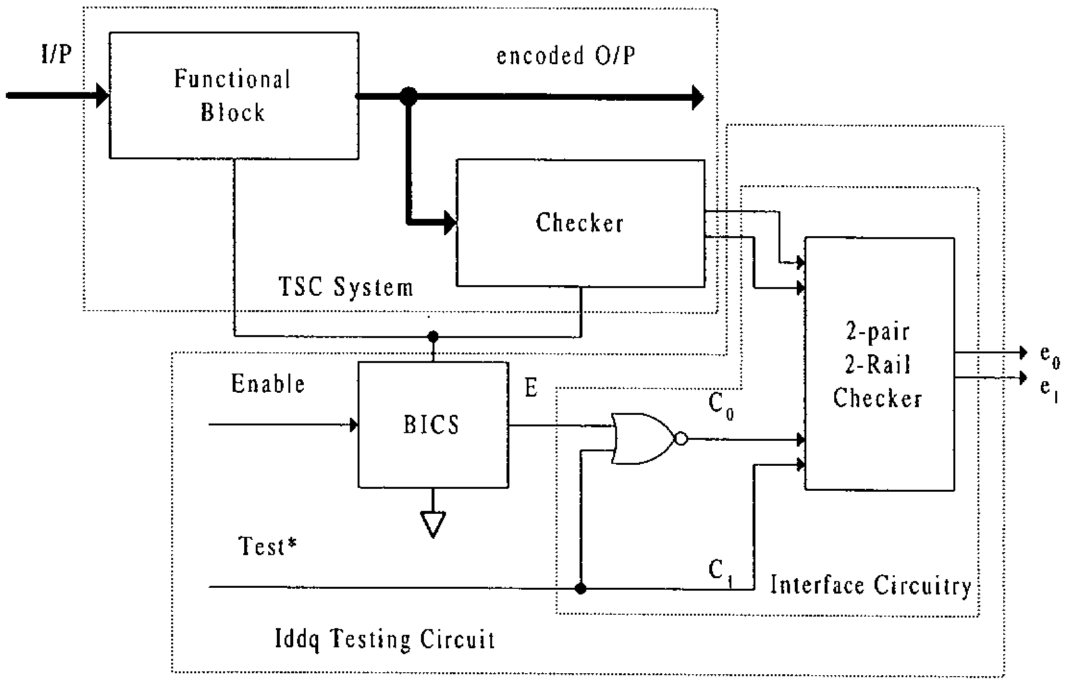


**Figure 5.7** Additional testing circuitry for Iddq testing in TSC circuit.

**Table 5.2** Summary of the operation for Iddq testing circuit.

| E | Test* | $C_0$ | $C_1$ | operation mode |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | fault-free |
| 0 | 1 | 0 | 1 | fault detection is disabled |
| 1 | 0 | 0 | 0 | fault is detected |
| 1 | 1 | 0 | 1 | fault detection is disabled |

*B. Fault Analysis of the Iddq Testing Circuit*

The circuit structure of a CMOS NOR gate is shown in Figure 5.8(b). The NOR gate is an n-dominant CMOS network. As shown in Table 5.3, under the occurrence of a single transistor stuck-on fault, the CMOS network will become a potential divider and an indeterminate logic value will be produced. The value of the output voltage depends on the resistance of the transistors, which is equal to ( Rn / ( Rn + Rp x 2 ) ) x Vdd. Based on the design rules proposed in [MILL 91], by increasing the resistance of the PMOS transistor, while keeping the NMOS unchanged, the indeterminate output voltage will fall into a determinate logic value. The indeterminate logic value X denoted in Table 5.3 can now be treated as logic 0. For the 2-input NOR used in the interface circuitry, the occurrence of transistor stuck-on fault at N1 or N2 is the same as output stuck-at 0, which can be detected by means of non-code word output at ($C_0$, $C_1$). Transistor stuck-on fault at P1 and P2 will not affect the operation of the current testing circuit. Therefore, from the logical point of view, a two-rail checker implemented with NOR gates can be regarded as strongly code-disjoint with respect to transistor stuck-on fault.

**Table 5.3** Output of transistor stuck-on fault in CMOS NOR gate(of Figure 5.8(b)).

| A B | P1 | P2 | N1 | N2 | fault-free |
|---|---|---|---|---|---|
| 0 0 | 1 | 1 | X | X | 1 |
| 0 1 | 0 | X | 0 | 0 | 0 |
| 1 0 | X | 0 | 0 | 0 | 0 |
| 1 1 | 0 | 0 | 0 | 0 | 0 |

Detection of stuck-at fault at output E of BICS(shown in Figure 5.7) is dependent on the testable design of the BICS circuit. The BICS is an analog circuit which has analog input and digital output. We assume that any fault in the BICS will result in stuck-at fault at its output E. As shown in Table 5.2, line E stuck-at 0 cannot be detected by means of non-code word output at ($C_0$, $C_1$). However, it can be detected by periodically exercising the BICS by applying a large test current.

Line $C_0$ stuck-at 0 will mask the output E of the BICS, but it can be detected by applying Test*=0 and disabling the operation of the current sensor at the same time to ensure E=0. Similarly, stuck-at fault at line Test* can also be detected by disabling the current sensor and applying test signal at line Test*.

It should be noted here that the Iddq testing circuit may lose the ability to detect the steady current from the TSC circuit, but still produces code word 01 or 10 at ($C_o$, $C_1$). Since it is not part of the code words checking circuit (i.e. the checker), any malfunction of the Iddq testing circuit will not affect the normal operation of the TSC circuit.

### 5.2.4 Application Example

We now demonstrate the effectiveness of Iddq testing based on the proposed test generation method in Chapter 4. In practice, a 2-pair two-rail checker(TRC) may be composed of three different ways: 1) AND-OR, 2) NAND and 3) NOR gates implementations. The method developed in Chapter 4 has been proved to be valid for AND-OR type TRC. However, it is also applicable to NAND and NOR type TRCs.

Here, we consider the implementation of the TRC in the form of NOR gates structure as shown in Figure 5.8(a).

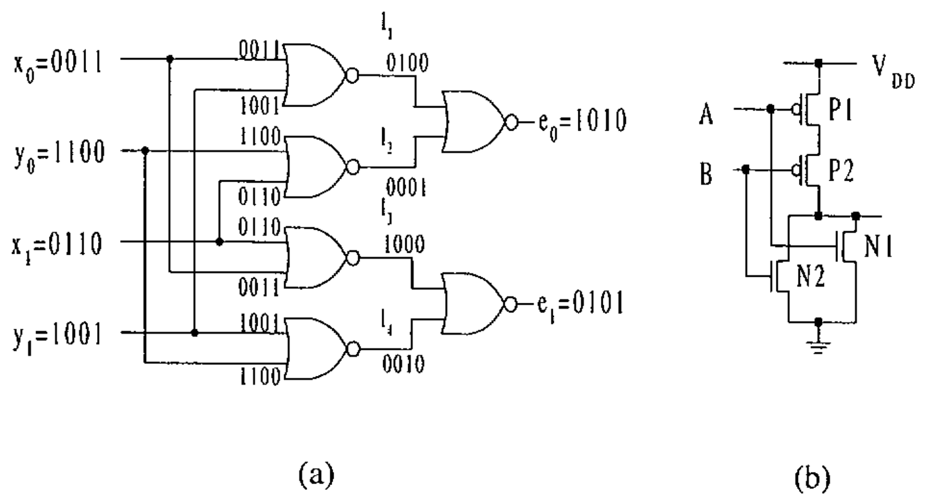


**Figure 5.8** A NOR-type 2-pair two-rail checker.

In the Iddq testing circuit design, the 2-pair TRC is designed to be n-dominant. However, we believe that this is not a good solution to solve the bridging fault coverage problem. Because by making the circuit n-dominant or p-dominant, the switching time will be affected. Therefore, the operation speed of the TRC will be degraded. This leaves much to be desired. Furthermore, a large number of bridging faults still cannot be detected purely by logic testing. Iddq testing in TRC circuit is still required. Next, we consider the bridging fault detection at the gate-level and transistor-level with the proposed test generation method.

### *A. Bridging Fault Detection at Gate-level*

In theory, bridging between any two adjacent lines is possible. Practically, the occurrence and detection of bridging faults depend on the circuit layout. In this study, we consider two types of bridging faults at the gate-level, namely, non-feedback bridging faults and feedback bridging faults. We assume that the adjacency of the interconnections is maintained at the layout level.

Non-feedback bridging fault includes bridging faults in which the bridged lines are exclusive, for example, faults F1 to F3 in Figure 5.9 are of this type. Whereas, bridging fault F4 in Figure 5.9 is known as feedback bridging fault. Under the occurrence of feedback bridging fault, the behaviour of combinational circuit will become asynchronous sequential circuit due to the feedback loop. Apart from those bridging faults indicated in Figure 5.9, bridging fault can occur within line $l_1$ to $l_4$ in Figure 5.8(a) and also between these lines and any line of any block.

Iddq testing is effective only when the logic values applied on the two bridged lines are different. Under this condition, a conducting path from $V_{DD}$ to GND is established to carry extra large power supply current. Fault is then detected by the current sensor.

Firstly, let us consider the non-feedback bridging fault coverage problem. We summarise the analysis as follows.

1. Bridging occurred within any input pair, say fault F1. Since the test vectors applied on any input pair are code words (01 or 10), fault must then be detected.

2. Similarly, the bridging with line $l_1$ to $l_4$ can also be detected since those lines can only receive code words (i.e. 1-out-of-4 code).

3. Detection of F2 and F3 can be done in the same way. If all lines carried different bit sequences through the test, there must be at least one bit difference between any two lines. Then, bridging fault can be activated and be detected.

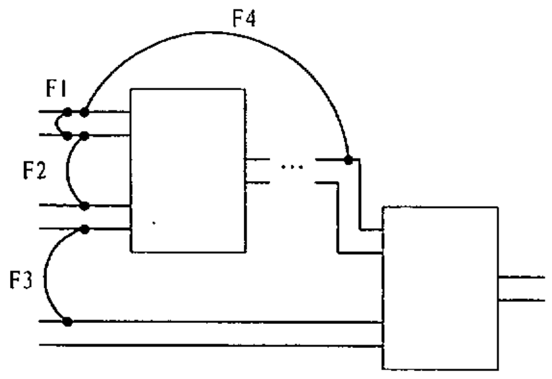


**Figure 5.9** Bridging faults at gate-level.

Secondly, feedback bridging fault may introduce oscillation at the output depending on the number of inversions on the path. If the number of inversions is odd, the circuit will oscillate. Otherwise, it will not. Let us consider the feedback bridging fault in a 2-pair TRC such as line $x_0$ and line $l_1$ in Figure 5.8(a). Simulation showed that any bridging fault in a two-rail checker will result in high power supply current dissipation when two bridged lines have different values.

Our test set is very efficient in detecting both feedback and non-feedback faults at gate-level in Iddq test. As shown in Figure 5.9, every line will carry different test sequence. There is at least one bit difference between any two lines. As mentioned previously, bridging between any two lines would result in high power supply current dissipation in both feedback and non-feedback bridging when the two lines have different values. So, those faults can be detected in Iddq test.

*B. Intratransistor Bridging Fault Detection*

In the intratransistor bridging fault model, we consider the bridging between drain, source, gate and bulk nodes of a transistor. Table 5.4 summarises the analysis of this fault in a CMOS NOR gate as shown in Figure 5.8(b). It is found that three input test vectors (00, 01, 10) are sufficient to activate any transistor bridging fault in a CMOS NOR gate to generate abnormal Iddq and produce the non-code word (0,0) at ($C_0$, $C_1$).

Since TRCs are code checkers, only codewords are available in the normal condition. It is necessary to make sure that enough codewords can be produced, so that the three test vectors will appear at the input of each NOR gate in the checker. Using the test generation method described in Chapter 4, we can see that all four code words (0101, 0110, 1010, 1001) are presented at the input of each block of the TRC as shown in Figure 5.8(a). These four codewords are enough to activate any intratransistor bridging fault in the TRCs implemented by NOR gates. Abnormal Iddq current is then detected by the BICS and results in non codeword (00) being generated at the interface.

In summary, by implementing Iddq testing technique into CMOS static logic circuits, the overall testability can be greatly enhanced.

**Table 5.4** Iddq in static CMOS NOR gate(of Fig. 5.8(b)) under intratransistor bridging faults.

| | P1 | | | | | | P2 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AB | sd | gs | gd | sb | gb | db | sd | gs | gd | sb | gb | db |
| 00 | L | H | H | L | H | L | L | H | H | L | H | L |
| 01 | L | H | H | L | H | L | H | L | H | L | L | H |
| 10 | H | L | H | L | L | H | L | L | L | H | H | H |
| 11 | L | L | L | L | L | L | L | L | H | L | L | H |
| | N1 | | | | | | N2 | | | | | |
| 00 | H | L | H | L | L | H | H | L | H | L | L | H |
| 01 | L | L | L | L | L | L | L | H | H | L | H | L |
| 10 | L | H | H | L | H | L | L | L | L | L | L | L |
| 11 | L | H | H | L | H | L | L | H | H | L | H | L |

( H - high current, L - same as fault-free ones)

## 5.3 TESTABILITY ANALYSIS OF DOMINO-CMOS LOGIC CIRCUITS

In this section, we study the testability analysis of domino-CMOS logic circuits. CMOS logic circuits can be roughly classified as static CMOS and dynamic CMOS logic circuits. A CMOS static gate consists of two complementary NMOS and PMOS logic networks, connecting the output node to ground(GND) and $V_{DD}$ respectively. Dynamic gates require extra clock to first pre-charge(pre-discharge) the circuit and then evaluate the n(p)-logic block [SHOJ 87]. Domino-CMOS gates, which have certain kinds of advantages than other dynamic gates, play an important role in the design of dynamic logic.

An example of domino-CMOS logic circuit is shown in Figure 5.10. The operation of a domino-CMOS gate consists of two phases. In pre-charge phase, transistor P1(N2) is turn on(off) so that node Vout* is charged to logic 1. Input is then applied and becomes stable in the evaluation phase. A feedback transistor Pf is carefully designed to reduce the problem of charge sharing which may result in loss of noise margin and even create a logic error. In [JHA 84][JHA 88][JHA 90], it was shown that transistor stuck-on and stuck-open faults in the n-logic network can be mapped as stuck-at 1 and stuck-at 0 faults at the corresponding input node in the gate-level circuit. Also, stuck-open fault in the transistors P1, N1, P2, N2 can be mapped as either stuck-at 0 or stuck-at 1 at the output. Transistor stuck-on fault in these four transistors can be detected or has no effect in logic testing if the circuit is made p-dominant or n-dominant. The above observation is based on the assumption that the faulty stuck-on transistor has the same conductance as a fault-free one. But this assumption is often unrealistic [METR 95]. The stuck-on transistor may have higher resistance value though it can still allow current flow regardless of the input data. If the stuck-on transistor is included in the conduction path in the dynamic network, the pre-charging/discharging process will be affected and the output of the gate will be degraded. Similarly, stuck-on fault in the static inverter will also degrade the output waveform and reduce the noise margin.

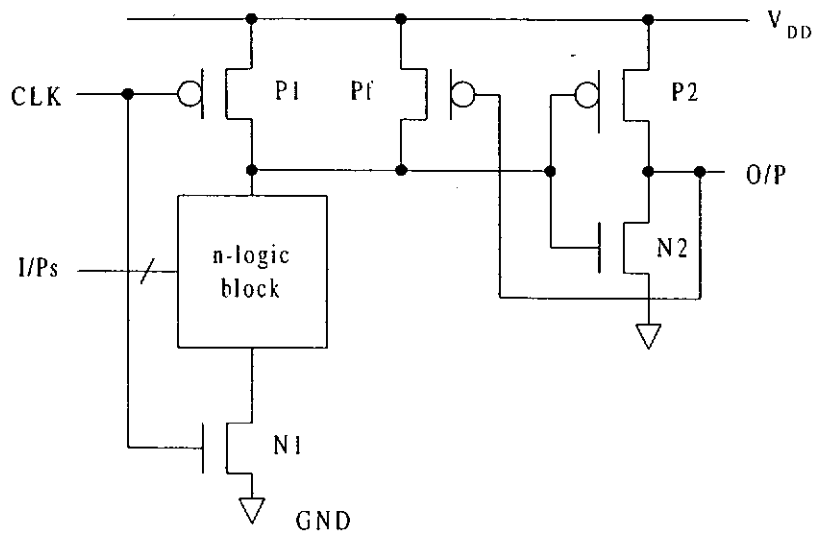
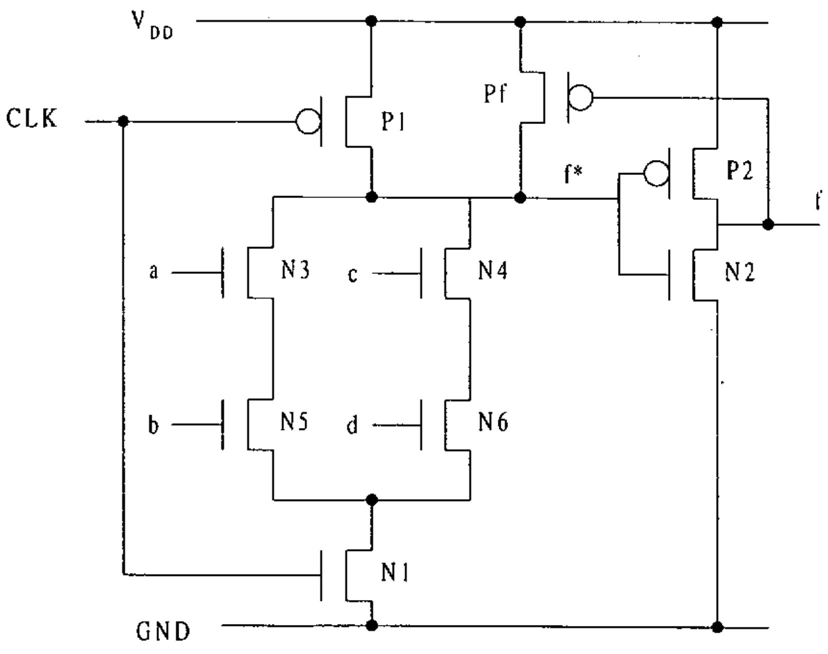


**Figure 5.10** A domino-CMOS logic circuit.

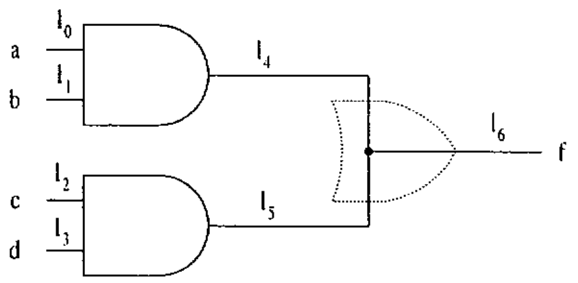
A bridging fault between any two internal nodes is possible. To analyse and generate test vectors for each bridging fault is quite a complex task. Instead, in our fault model, we consider only the most likely occurred bridging faults. These faults are called intratransistor bridging faults which include drain-gate, drain-source, gate-source bridging and gate-level bridging faults.

### 5.3.1 Iddq Testability

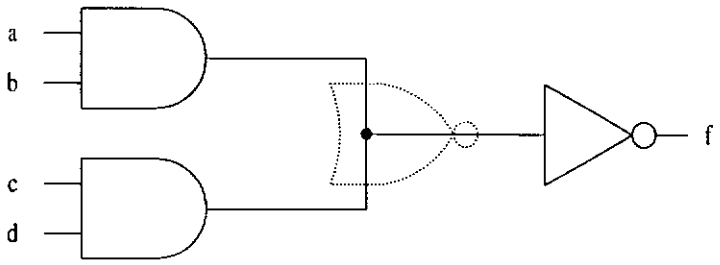
In this section, we use Figure 5.11 as an example to study the Iddq testability. The faults we considered in this study are intratransistor bridging faults and all the testable bridging faults. Sometimes, transistor stuck-on fault are also known as bridging between drain and source nodes of MOS transistor [RAJS 95] if the bridging resistance is small comparing to the resistance of a turned on transistor.



(a)



(b)



(c)

**Figure 5.11** (a) Domino-CMOS logic circuit, (b) gate-level representation, and (c) modified gate-level representation.

*A. Detection of Intratransistor Bridging Faults*

To detect a bridging between two electrical nodes in the circuit, we have to activate it by setting the two nodes in opposite logic values. But it is very complicated to examine all possible bridging between two nodes because the fault set is very large. Instead, we can use the stuck-at fault test vectors which are developed by many ATPGs to determine the fault coverage. Table 5.5 summarises the required test vectors for the detection of all single stuck-at faults as shown in Figure 5.11(b). Four test vectors are enough to detect all single stuck-at faults based on logic testing approach.

**Table 5.5** Test vectors for circuit Figure 5.11(b) based on logic testing.

| Test vector [ a b c d ] | Stuck-at fault |
|---|---|
| T1 [ 1 1 0 0 ] | $l_4$ s-at 0, $l_6$ s-at 0 |
| T2 [ 0 0 1 1 ] | $l_5$ s-at 0, $l_6$ s-at 0 |
| T3 [ 1 0 0 1 ] | $l_1$ s-at 1, $l_2$ s-at 1, $l_6$ s-at 1 |
| T4 [ 0 1 1 0 ] | $l_0$ s-at 1, $l_3$ s-at 1, $l_6$ s-at 1 |

In Table 5.6, we summarise the intratransistor bridging fault coverage of Figure 5.11(a) by the stuck-at fault test vectors in Table 5.5. It is shown that most intratransistor bridging faults except those drain-source bridging in transistor N1, N3, N4, N5, N6 can be detected by this test set. The undetected faults are due to the fact that P1 is off during the evaluation phase. And also, during the pre-charge phase, neither n-logic block nor N1 is turned on. There is no conducting path from $V_{DD}$ to GND node to carry extra large faulty current unless multiple faults occur.

**Table 5.6** Test results of intratransistor bridging faults of Figure 5.11(a) based on Iddq testing.

| Test vector | Detected faults |
|---|---|
| CLK=0 | P1(gs, dg), P2(dg, ds), P3(dg, gs), N2(dg, gs) |
| CLK=1 | N1(gs, dg) |
| T1 | P1(dg, ds), P2(ds, gs), P3(ds, dg), N2(dg, ds), N3(dg, gs), N5(dg, gs) |
| T2 | P1(dg, ds), P2(ds, gs), P3(ds, dg), N2(dg, ds), N4(dg, gs), N6(dg, gs) |
| T3 | N6(dg, gs) |
| T4 | N5(dg, gs) |

The drain-source bridging faults of the NMOS transistors of n-logic block cannot be detected by current testing. Transistors P1 and N1 are turned on exclusively during normal operation. No conducting path from $V_{DD}$ to GND will be formed. Since the bridging may not be perfectly short circuit, the resistive value might be varied from less than 500Ω to greater than 20KΩ as reported in [RODR 96]. In order to estimate the effect of the bridging resistance to the output waveform of the domino-CMOS circuit, different bridging resistors are inserted in parallel with the affected NMOS transistor(for example, transistor N5) in Figure 5.11(a). Then suitable test vector is applied to the circuit such that no conducting path in the n-logic network will be formed. In normal fault-free condition, the output of the gate will maintain logic 0. However, if there is a bridging between drain-source of an off transistor, a conducting path is formed. The circuit will discharge and the output of the circuit will toggle. Figure 5.12 illustrates the simulation result obtained from the PSPICE electrical circuit simulator.

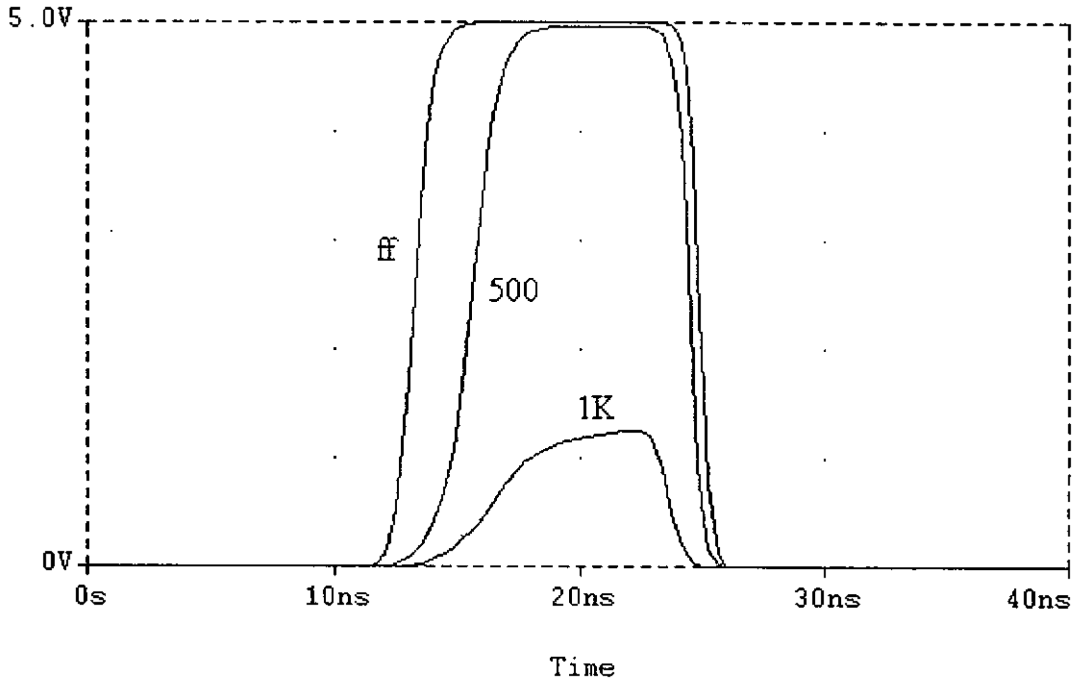


**Figure 5.12** Output waveforms under different drain-source bridging resistance.

For small bridging resistance value, the node f* will be discharged very quickly and the output will switch to logic 1. This logic error will propagate to the primary output and be detected as logical fault. On the other hand, the output will degrade gradually as the bridging resistance increases. Based on this observation, the drain-source bridging fault can be modelled as stuck-at 1 fault on the corresponding input of the 2-input AND gate for a range of bridging resistance values. Thus this fault can be detected by stuck-at fault test set. Depending on the complexity of the circuit, the degradation of output waveform may or may not result in logic error.

*B. Detection of Gate-level Bridging Faults*

Gate-level bridging faults model is another popular method to describe the bridging between any two logical nodes in the circuit. Unlike the intratransistor bridging fault model which considers each transistor individually, it provides a global view of the bridging fault in the circuit. Since the occurrence of bridging fault is layout and implementation dependent, we have to consider the mapping between the logic circuit and the implementation. Again, let us consider the circuit shown in Figure 5.11(a) as an example. It mainly consists of two parts: a dynamic functional circuit and a static inverter. In gate-level implementation, it is represented by AND-OR logic circuit. There are two drawbacks in this type of mapping. The OR gate virtually exists in the transistor-level, which performs wired-OR function. Any node bridged with the input of the OR gate can be mapped to a bridge with the output of the OR gate. Another drawback is that the static inverter in the transistor implementation does not exist in the gate-level representation. In order to have higher real fault coverage, we have to modify the logical representation to another form as shown in Figure 5.11(c). The OR gate is replaced by the NOR and inverter gates. The wired-OR function is now represented by the wired-NOR. Physically, we now have three types of circuit nodes that can be bridged together. They are (a) input of AND gate, (b) input of inverter gate, and (c) output of inverter gate.

Unlike other dynamic logic, domino-CMOS logic always has a static inverter at the network output to drive other gates. To detect any bridging fault of type (a) and type (c), we can activate this fault by setting these two nodes in complement logic values. If the node of type (b) is bridged with the node of type (a) or (c), we can activate this fault by setting the dynamic node (b) to logic 0 and the other node to logic 1. As a result, the dynamic node is connected to ground node and the node of type (a) or (c) is effectively connect to $V_{DD}$. When type (b) node is bridged with another node of logic 0, only very little discharge current will flow through the circuit. The discharging process is dependent on the resistance of the bridging. It may or may not provide a logical error at the output. If the discharging is fast enough compared to the clock rate, a logical error should be detected at the output. However, this bridging fault can be reduced by designing the layout carefully. Based on these observations, we can examine the coverage of the gate-level bridging faults by the stuck-at fault test set generated by ATPG.

### 5.3.2 Testability Improvements

As discuss in previous paragraph, some bridging faults cannot be detected by current testing. Under certain conditions, the drain-source bridging of the transistors in the n-logic block can be modelled as logical fault and detected by logic testing. However, it can be shown that the bridging fault in transistor N1 cannot be detected by both logic and current testing. And also, the existence of undetectable fault may mask other faults in the circuit. The inherent structure of domino-CMOS logic circuits prevents the effective use of Iddq testing method. Therefore, modification of the domino-CMOS logic gate is required in order to facilitate the Iddq testing. Two important factors are considered: a) the design should be simple, and b) the additional circuitry must be easily testable. Based on these criteria, we modified the domino-CMOS gate by the following two schemes.

*A. Scheme 1*

The first scheme improves the testability by adding an extra transistor in each gate for controllability. The resultant design is very simple and easily testable. As shown in Figure 5.13, only one additional transistor is required for each gate to improve the fault coverage. We now explain how the modified domino-logic gate works.

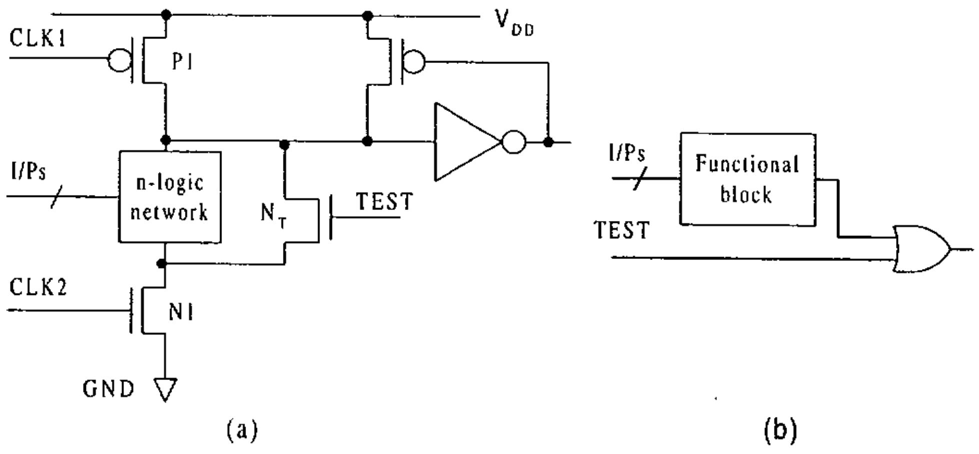


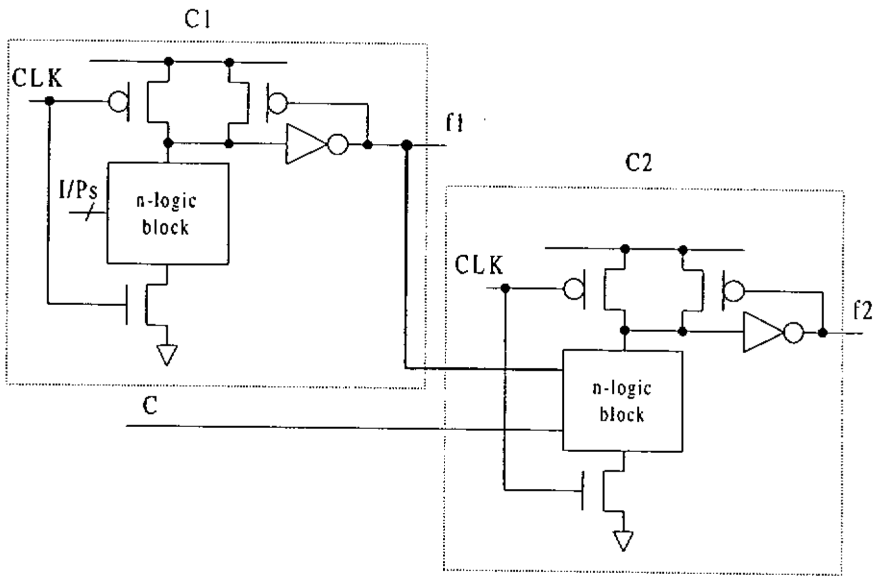
**Figure 5.13** (a) Modified domino-CMOS gate, and (b) equivalent gate-level representation.

In normal mode, TEST is set to logic 0 to turn off transistor $N_T$. The two clock inputs, CLK1 and CLK2, are set to the same logic value. The operation of the modified domino-CMOS logic gate is the same as unmodified one. In test mode, the drain-source bridging fault of transistor N1 cannot be detected in both pre-charge phase and evaluation phase by either logic or current testing. So to detect transistor N1 drain-source bridging, P1 and $N_T$ are turned on. For fault-free case, only leakage current can flow in the circuit since N1 is normally turned off. The output of BICS should indicate no error. If drain-source nodes of N1 is bridged, a large current will flow from $V_{DD}$ to GND, this fault is then detected by the BICS. Apart from this fault, P1(dg, gs), N1(dg, ds) will result in abnormal high current flowing through the circuit and will also be detected by the BICS. The test has to be applied in the pre-charge phase, since this test will affect the pre-charged nodes. An additional clock for this test is required.

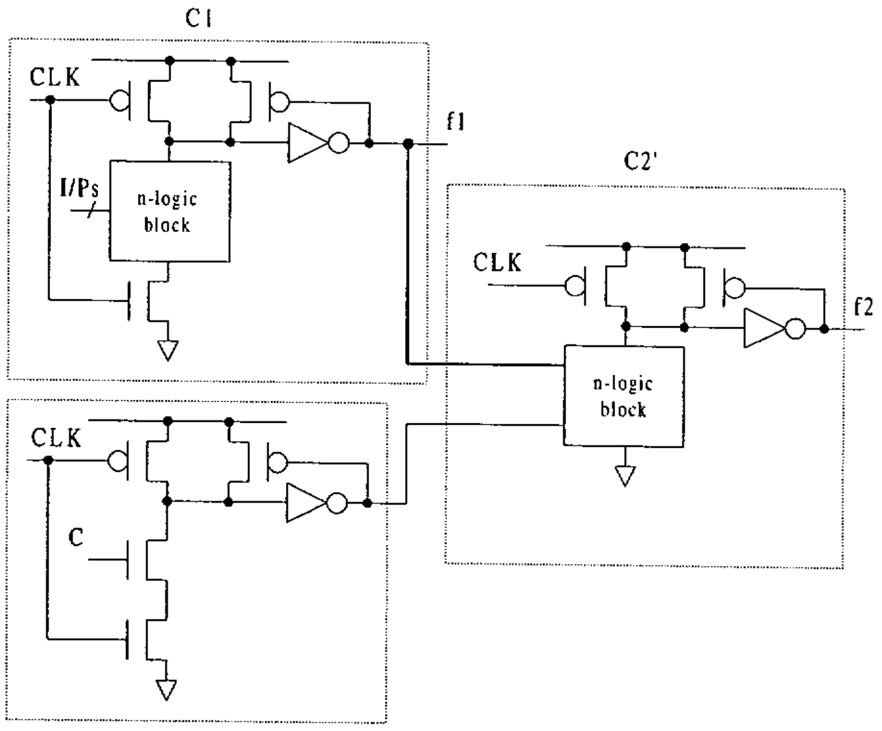
After discussing testability improvement of the modified domino-CMOS logic gate, we now consider the fault in the additional transistor $N_T$. $N_T$ gate-source and drain-gate bridging faults can be detected by setting TEST=1 during any evaluation phase. By turning on $P_1$ at the evaluation phase, the $N_T$ drain-source bridging can be detected if the transistor is normally off. However, a break at $N_T$ is undetectable by neither Iddq testing or voltage testing. The occurrence of these faults could not affect the normal operation of the circuit, but would contaminate the testing improvement scheme.

*B. Scheme 2*

The second proposed scheme is based on the observation that the input signals of any gate, except the primary input signal, would not change before the evaluation phase. No conducting path will be formed to discharge the circuit. So, transistor N1 of Figure 5.11(a) is not necessary. An example is shown in Figure 5.14, the modification is quite simple. A dynamic buffer is inserted in primary input line C and sub-circuit C2. Transistor N2 of C2 is removed. Based on this scheme, the inputs of an n-logic block can be either all primary input or all non-primary input. Bridging between drain and source of N1 can be detected by suitably setting a test vector at the primary input during pre-charge phase.



(a)



(b)

**Figure 5.14** Example of domino-logic circuit, (a) original, and (b) modified one.

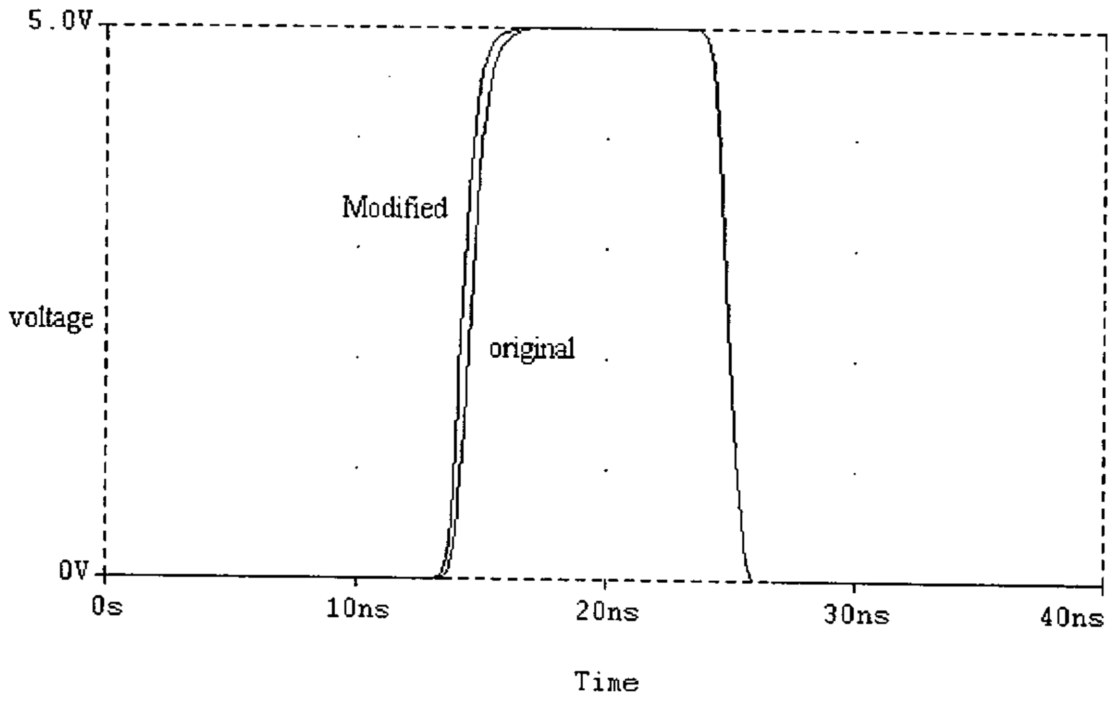


**Figure 5.15** Comparison of output waveforms of original domino-CMOS gate and modified one of Figure 5.11(a).

We studied the effect of removing N2 in the domino circuit. Simulation result is shown in Figure 5.15. With all parameters kept constant, the evaluation speed is in fact faster than the original one because N1 is removed from the discharging path. The testability of the modified gate is improved. Since the gate which is controlled by primary inputs has transistor N1, any drain-source bridging of N1 is detectable by Iddq testing in the precharge phase because a conducting path can be created in the n-logic block by suitable input vectors. Despite of this advantage, this modification scheme requires more overhead than scheme 1.

## 5.4 SUMMARY

Testability analysis and a combined Iddq and logic testing approach of static and domino-CMOS logic circuits are studied in this chapter. In fact, Iddq testing technique is essential in these two design methods because it has the advantage of easy test generation and the ability of detecting many undetectable faults (mainly bridging fault in a purely voltage testing environment). Later, we presented two testability improvement schemes for domino-CMOS logic circuit such that Iddq testing is more applicable to it. Though these schemes require extra overhead, the testability improvement overweighs the cost. Comparing the static and domino-CMOS logic circuits, domino-CMOS logic circuit has the advantage of static-CMOS, which is low power consumption. But the testability of domino logic is much better than its static family as discussed in previous paragraphs because many faults such as stuck-on and stuck-open faults can be mapped to stuck-at faults. So, domino-CMOS logic design is more applicable to self-checking design. However, domino-CMOS logic requires more precise operation and is more easily influenced by radiation and noise environment.

# Chapter 6

# Built-in Intermediate Voltage Sensor

## 6.1 INTRODUCTION

In previous chapters, we have focused on the study of off-line testing technique for multiple stuck-at faults and bridging faults. The objective of studying these testing methods is that the design problem of totally self-checking(TSC) circuits can be solved in the case of single logical fault [ABRA 90][ADER 73][BURN 94][BUSA 94]. However, large number of defects can occur when the CMOS VLSI chips are in operation mode. Some of those defects can have effects similar to bridging faults in CMOS circuits. That is, indeterminate logic level can arise at the faulty sites.

In Chapter 5, the effects of bridging faults in CMOS logic circuits are studied. It has been shown that bridging faults can create a conducting path between $V_{DD}$ and ground node. A potential divider network is formed such that the output of the network will fall in between $V_{DD}$ and ground node. In this respect, the bridging fault detection problem becomes more complex because of the indeterminate logic values at the faulty sites. The detection of these faults by logic testing depends on the level of the intermediate voltage at the output of the faulty gate with respect to the threshold voltage of the inputs of the fan-out gates. Thus, these faults may not result in logical error, but may significantly affect the circuit timing.

Iddq testing was reported to be very effective to detect the bridging faults [LEVI 81][MALY 92][MIUR 92][RAHS 95][STOR 90][STOR 91]. However, supply current monitoring is relatively slow and requires very precise current monitoring device. Generally, a current sensor compares the measured current with the predefined reference current. If the measured current is larger than the reference current, error will be indicated. However, the speed of current measuring is dependent on the resolution of the current magnitude. The higher the resolution, the slower is the detection speed. On the other hand, high speed detection may result in poor testing quality. So far, this technique is only applied to fully static CMOS circuits. Furthermore, it is not suitable for on-line testing approach.

Recently, [TANG 95] proposed to use intermediate voltage testing technique to detect the bridging faults. It is based on the observation that a large number of bridging faults can still be detected by logic testing [STOR 90]. The faults that cannot be detected by logic testing are mainly those faults that can result in indeterminate logic values at the fault sites. A sensing circuit called built-in intermediate voltage sensor(BIVS) was proposed to detect these faults. Three designs of BIVS were given. The drawback of these BIVS designs is twofold: the large power consumption and lack of testability consideration. So, their designs are not suitable for TSC application.

A plain example can show that it is essential to detect the occurrence of indeterminate logic values in the circuit. If the output of the functional block is within the indeterminate logical region, the checker may interpret the output as a codeword, while other functional blocks may consider it as a non-codeword. As a consequence, the objective of implementing a self-checking system can no longer be achieved.

Based on this observation, we propose a new and novel static CMOS BIVS design. Detailed analysis shows that our sensing circuit is not only capable of detecting the intermediate voltage at its input, but also can detect or tolerate the fault in itself. So, our design is well suited to on-line testing.

## 6.2 DESIGN OF SENSING CIRCUIT

Firstly, let us introduce some terminology used and precisely define the logical regions in CMOS digital circuits. Generally, in logic systems, variables and circuits can be in one of the two states which are logic 0 and logic 1. In practise, the voltage level can be divided into three regions which are logical 0 region, logical 1 region and indeterminate logical region(denoted as logic X). The logical 0(1) input voltage range is always wider than the logical 0(1) output voltage range. The difference is called noise margin. It allows a certain noise voltage existing at the input of a gate which will not affect the input values interpretation of the gate. Figure 6.1 shows the definitions of the logical region and the noise margin. In this thesis, the indeterminate logical region is defined as the difference between the maximum input voltage of logic 0 (1.5V) and the minimum input voltage of logic 1 (3.5V) of a CMOS gate respectively. In the rest of this chapter, we denote the maximum input logic 0 voltage as $V_{Lmax}$ and minimum input logic 1 voltage as $V_{Hmin}$. And we shall refer to the intermediate voltage as any voltage value within the indeterminate logical region.

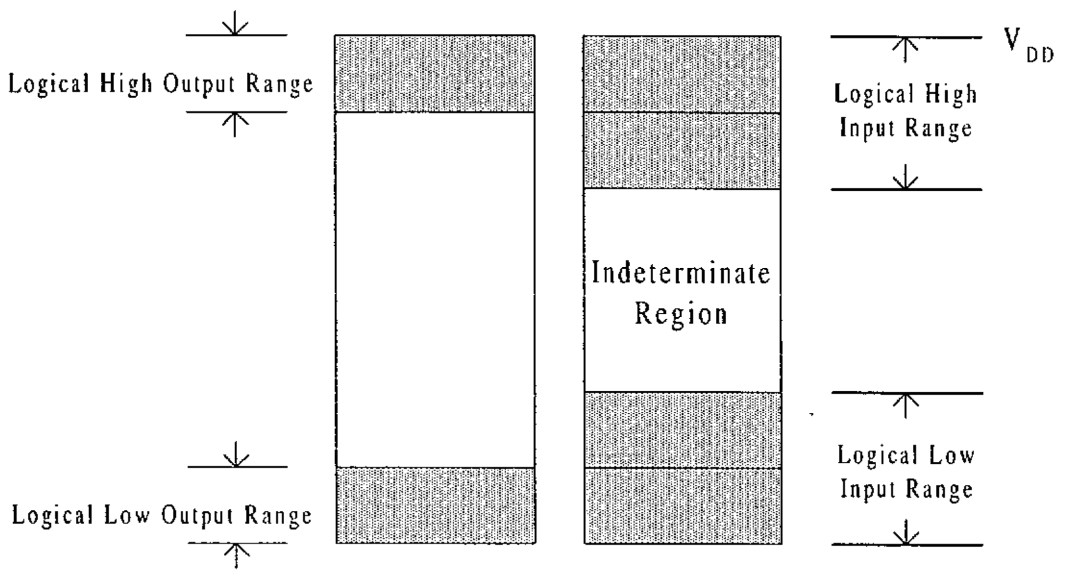


**Figure 6.1** Definitions of input and output logical range.

In the following paragraphs, we consider a novel design of a static CMOS built-in intermediate voltage sensor(BIVS). As discussed in previous paragraph, the logical region is divided into three level logic (0, X, 1). The proposed BIVS is used to detect the logic X on the line. The circuit is designed by a 2μm design rule. (Circuit simulation is performed using PSPICE level 3 electrical circuit simulator with $V_{DD}$=5V and Temp.=300K.) Figure 6.2 shows the proposed testing circuit. The BIVS consists of three basic parts: 1) a level shifter, 2) a switching network and 3) a simple current sensor. Table 6.1 denotes the sizes of the transistors.

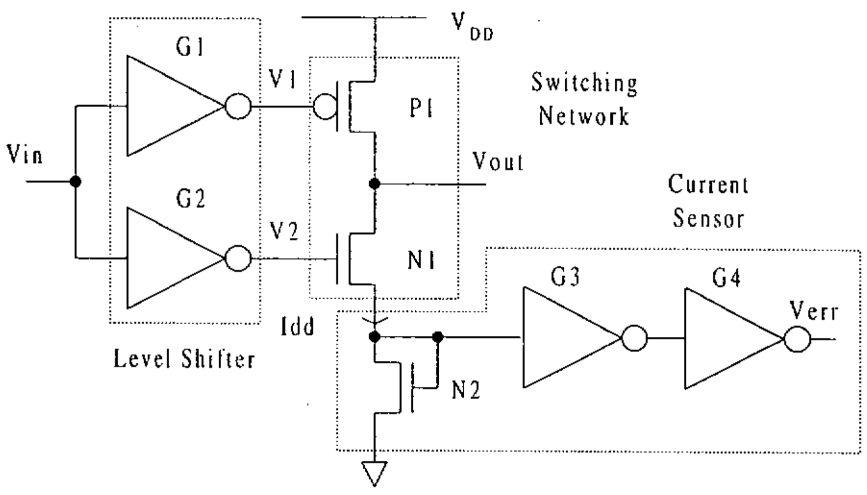


**Figure 6.2** Built-in intermediate voltage sensor(BIVS).

**Table 6.1** Summary of transistor sizes.

| Transistor | Size(W/L) in μm |
|---|---|
| $P_{G1}$ | 2/16 |
| $N_{G1}$ | 2/2 |
| $P_{G2}$ | 2/2 |
| $N_{G2}$ | 2/16 |
| P1 | 8/2 |
| N1 | 4/2 |
| N2 | 8/2 |
| $P_{G3}$ | 2/2 |
| $N_{G3}$ | 8/2 |
| $P_{G4}$ | 8/2 |
| $N_{G4}$ | 4/2 |

*A. Level Shifter*

The first part of the BIVS is a level shifter which consists of two inverters (G1and G2) of different transfer characteristics. The operation of a CMOS inverter is summarised in Table 6.2 and Figure 6.3 illustrates its DC transfer characteristic and operating region. The DC transfer characteristic of CMOS gate is controlled by the ratio of $\beta n/\beta p$, where $\beta = \frac{\mu\varepsilon}{tox}\left(\frac{W}{L}\right)$ is the MOS transistor gain factor [WEST 93]. For a given process, the effective surface mobility of the carriers in the channel $\mu$, the permittivity of the gate insulator $\varepsilon$ and the thickness of the gate insulator *tox* are process dependent, such that they cannot be altered easily. So, only the geometry dependent term (W/L) is changeable. Consider Figure 6.3, where the ratio of the gain factors is unity, so that the switching threshold is near half of the $V_{DD}$. If the ratio of $\beta n/\beta p$ is decreased, the switching threshold will shift from left to right. In order to distinguish the logic value at the input line of BIVS, the two inverters are designed such that their corresponding transfer characteristics labelled by V1, V2 are shown in Figure 6.4. When the input is logic 0(1), the two inverters produces same logic value 1(0). When the input falls in indeterminate logical region, the outputs of the two inverters will split into two different logic values. Based on this observation, the type of logical region presented at the input of the Level Shifter can be distinguished.

**Table 6.2** Summary of CMOS inverter operation.

| PMOS | NMOS | Output Region |
|---|---|---|
| non-saturated | cut-off | A |
| non-saturated | saturated | B |
| saturated | saturated | C |
| saturated | non-saturated | D |
| cut-off | non-saturated | E |

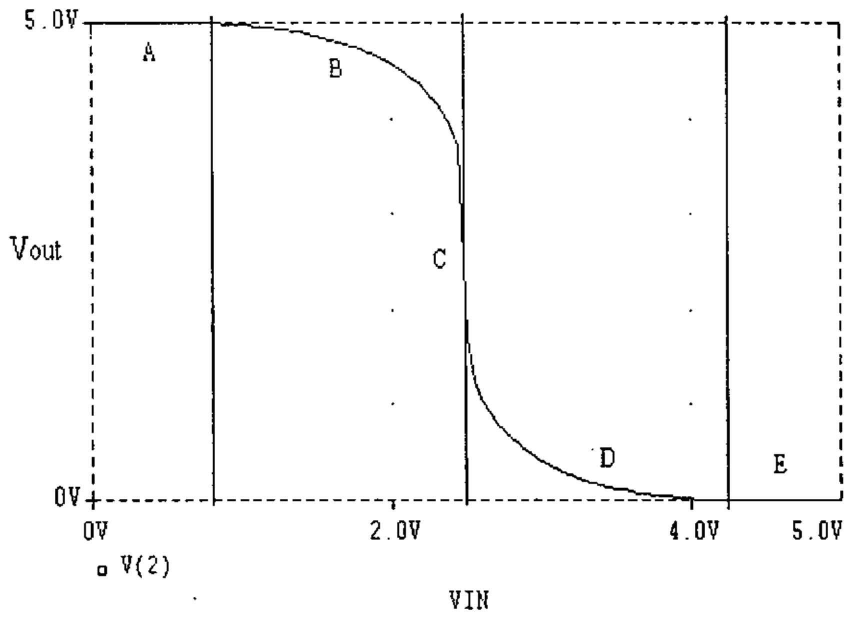


**Figure 6.3** CMOS inverter DC transfer characteristic and operating regions.

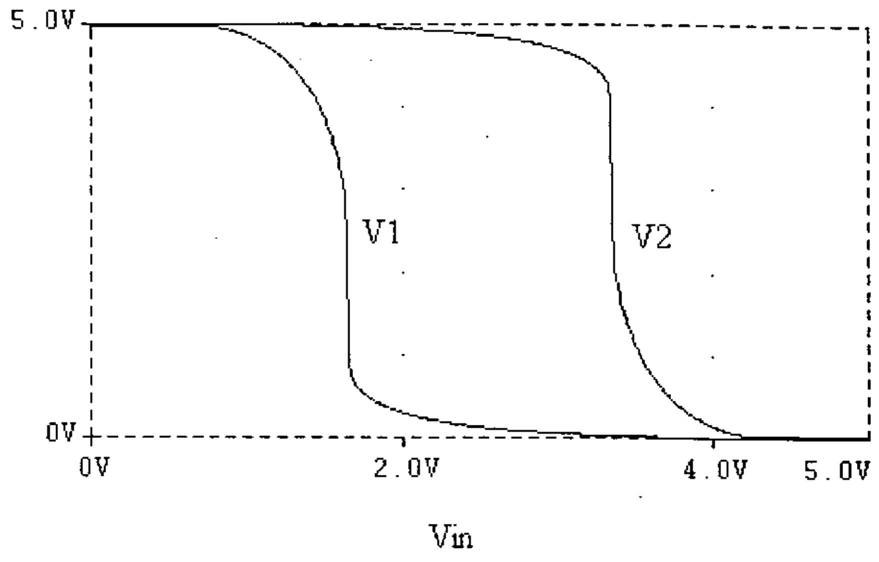


**Figure 6.4** Output voltage of inverters G1 and G2 respect to the input voltage.

*B. Switching Network*

The Switching Network is formed by a PMOS and an NMOS transistor which have the same gain factor. As discussed in the previous paragraph, when the Level Shifter receives logic 0(1) input, the outputs will be logic 1(0). Then either transistor P1 or N1(but not both) will be turned on and the Idd is very small. When the input voltage Vin falls into the indeterminate logical region, V1 will be at logic 0 and V2 will be at logic 1. Hence both P1 and N1 will be turned on and a large current Idd will flow through the switching network. The magnitude of the Idd current can be adjusted by changing the conductance of transistors P1 and N1.

*C. Current Sensor*

A simple current sensor, which is formed by N2 and two inverters(G3 and G4), is designed to detect the large Idd current. Generally, the current sensor performs a current to voltage conversion. During normal condition, the leakage current of the Switching Network is negligible. However, when both transistors of the Switching Network are turned on, a large voltage will be developed at the N2. Since a CMOS inverter is a high gain amplifier, the developed voltage will be amplified. A logic signal is produced at the output of the current sensor.

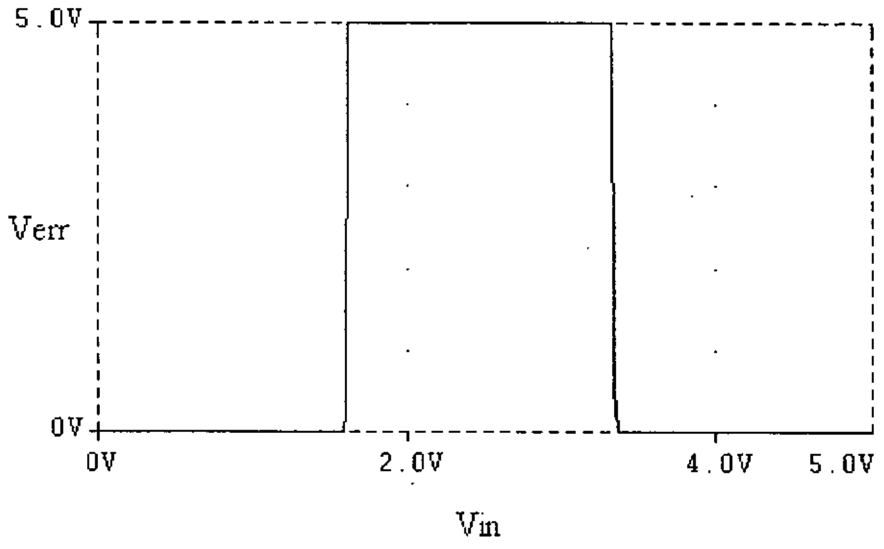


**Figure 6.5** Error output of BIVS respect to the input voltage.

Figure 6.5 shows the output of the BIVS respect to the input voltage. When the input voltage falls in between $V_{Lmax}$ and $V_{Hmin}$, both N1 and P1 will be turned on and a large Idd will flow through the network. The current sensor detects this large Idd current and produces a logic 1 at the output to indicate the error. In the design, we have obtained that $V_{Lmax}$ is 1.4V and $V_{hmin}$ is 3.3V.

## 6.3 TESTABILITY ANALYSIS OF BIVS SYSTEM

For highly reliable system design, it is important to verify that possible faults within the sensing circuit are detectable or that they will not affect the operation of the circuit. This is because these faults may mask the presence of the indeterminate logical error. In this section, we analyse the testability of the BIVS circuit under the presence of transistor stuck-on and stuck-open faults.

**Table 6.3** Behaviour of BIVS in the presence of transistor faults.

| Transistor | stuck-on | stuck-open |
|---|---|---|
| $P_{G1}$ | no effect | yes |
| $P_{G2}$ | yes | yes* |
| $P_{G3}$ | no effect | yes |
| $P_{G4}$ | yes | no |
| P1 | yes | yes* |
| $N_{G1}$ | yes | yes* |
| $N_{G2}$ | no effect | yes |
| $N_{G3}$ | yes | no |
| $N_{G4}$ | no effect | yes |
| N1 | yes | yes* |
| N2 | no effect | yes |

- yes* - fault effect is observed at Vout
- yes - fault effect is observed at Verr

Transistor stuck-on assumes that the faulty transistor is always at "ON" mode regardless of the voltages applied to the transistor. Table 6.3 summaries the detectability of these faults inside the BIVS. In Figure 6.2, inverter gates G1 to G4 are designed with different W/L ratios. Inverter G1 and G3 are designed as n-dominant, whereas inverter G2 and G4 are designed as p-dominant. So, if both the PMOS and NMOS network of an n-dominant inverter are turned on, the output voltage of the inverter will fall in the logic 0 region. Figure 6.6 demonstrates that the effect of transistor $P_{G1}$ is a stuck-on with respect to the input voltage Vin. On the other hand, an NMOS transistor stuck-on in a n-dominant inverter will result in an output stuck-at 0 fault. Similar analysis can be carried out in the p-dominate gates. Any transistor stuck-on fault in the Switching Network can be detected by normal input. It is easy to show that any transistor stuck-on fault in the BIVS can be detected at Vout or Verr, or has no effect on the operation of the BIVS.

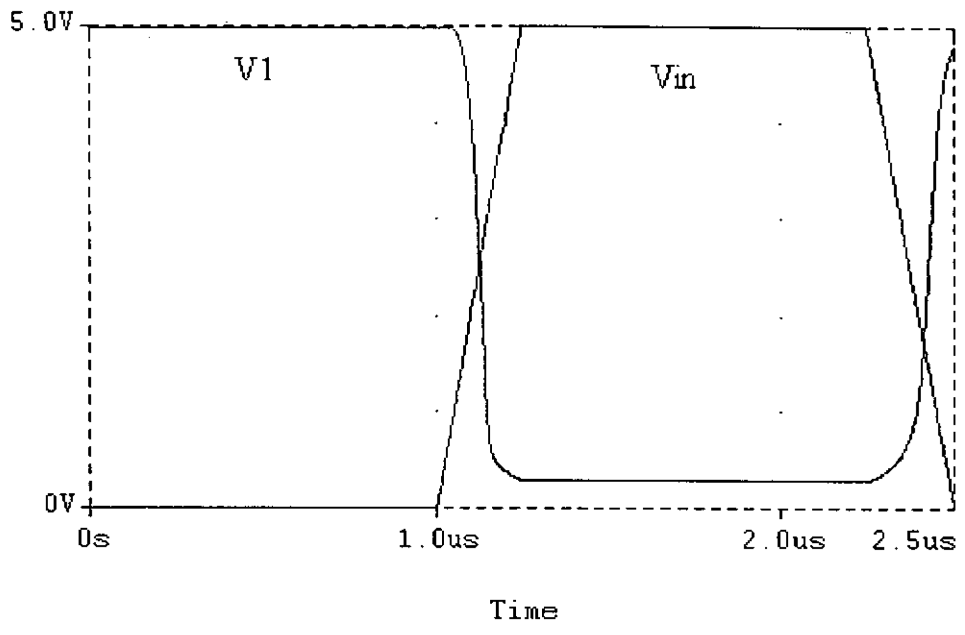


**Figure 6.6** Voltage V1 respect to the input voltage in under transistor $P_{G1}$ stuck-on.

Transistor stuck-open faults will prevent the conducting of a transistor such that the inverter will behave as a sequential circuit. If these faults occur within the Level Shifter and the Switching Network, logical error will result at the Vout node after the output of the corresponding faulty inverter gate has been charged or discharged. Therefore, these faults can be detected by the normal input. However, as listed in Table 6.3, stuck-open faults at transistor $P_{G4}$ and $N_{G3}$ cannot be detected during normal operation. These undetectable faults will prevent the detection of intermediate voltage of the BIVS. However, these faults can be flushed out by applying abnormal current input to the current sensor periodically.

Transistor N2 is designed as high resistive load device. Transistor stuck-on fault will have no effect on the operation of the BIVS. Since the leakage current of an

NMOS transistor is very small, the gate node of G3 will be charged up and the fault effect will then propagate to the output Verr of the BIVS. Nevertheless, this fault will not affect the detection of intermediate voltage presented at the input of the BIVS.

## 6.4 BIVS SYSTEM DESIGN AND ITS APPLICATION

### 6.4.1 BIVS System Design

In a self-checking system implemented with BIVS design, more than one BIVS will be used. An improved testable architecture of the BIVS system is built and is shown in Figure 6.7. In the design, a BIVS is inserted in the monitored line. As illustrated in Table 6.3, there are two undetectable transistor stuck-open faults. Although these two faults will not affect the normal operation of the circuit under test, they will mask other faults when multiple faults occur. However, this problem can be solved by our proposed design.

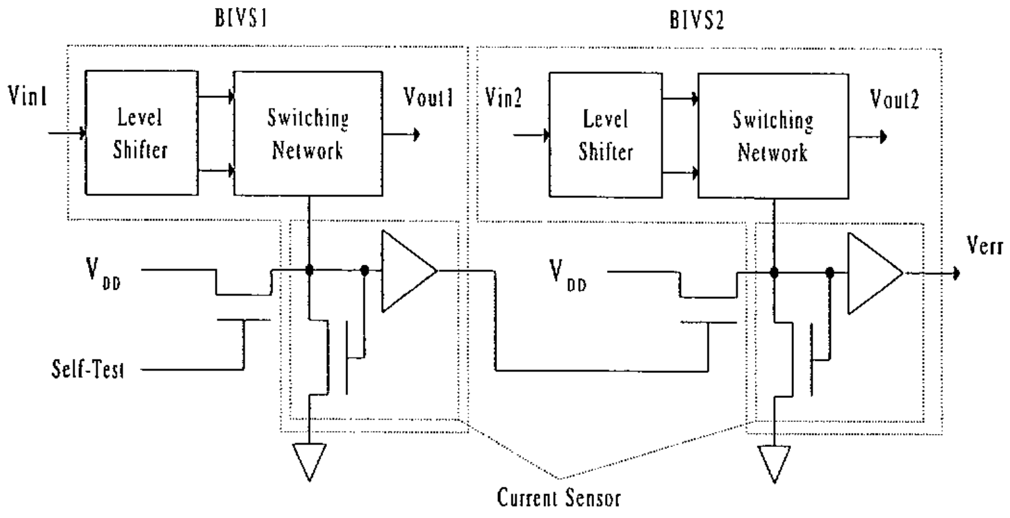


**Figure 6.7** BIVS system.

An additional transistor(as shown in Figure 6.7) is used to connect each BIVS block together serially. During the self-testing phase(Self-Test=1), a large current is applied to the left-most(BIVS1) of BIVS System. The current sensor will detect the abnormal current and produce a logic 1 at the output. This error signal will propagate to the last stage of BIVS system. Any undetectable transistor stuck-open fault(as shown in Table 6.3) will mask the error signal. And so, the transistor stuck-open faults can be detected. In addition, transistor stuck-on and stuck-open faults in the additional transistor can be detected during either normal operation or self-testing phase. So, the BIVS system is strongly code-disjoint(SCD) respect to transistor stuck-on and stuck-open faults based on the self-exercising concept [NICO 89a].

## 6.4.2 Integration of BIVS System with Totally Self-Checking Circuit

As discussed in Chapter 3, the effectiveness of the checker is dependent on the predefined fault model. However, when the output of the function block falls in indeterminate logic values, the checker may fail to detect this error. Example 6.1 further shows the effect of intermediate logic value at the output of the functional block.

**Example 6.1.** Assume that a functional block produces 3-out-of-5 codeword in normal operation. If a word 01x10 is generated at the output, the checker may interpret it as a codeword 01110, while other functional blocks may receive a non-codeword 01010. So, the erroneous output cannot be detected by the checker. Self-checking property is no longer validated. As a consequence, the normal operation of whole system is affected.

This problem can be solved by employing a BIVS as the output level detection element.

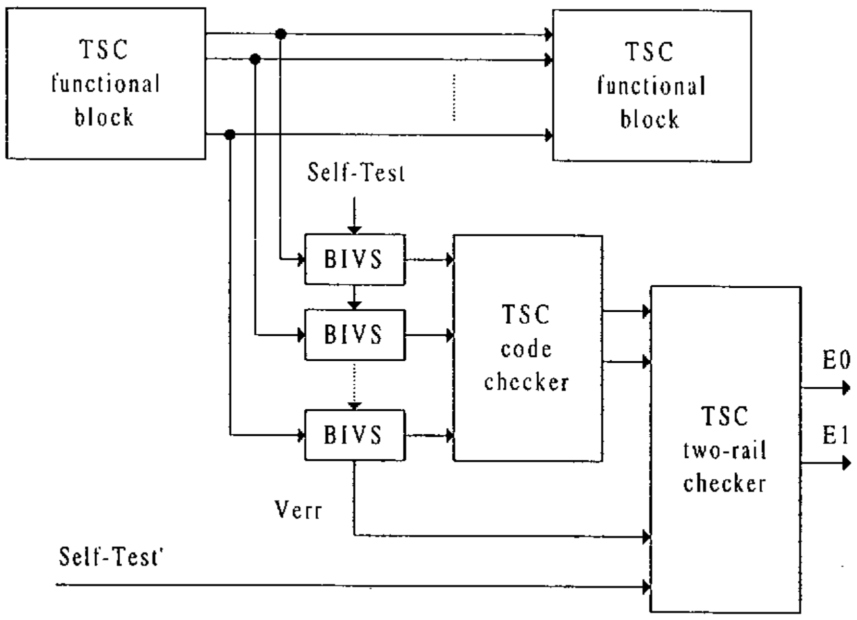


**Figure 6.8** Integrating of a BIVS system and a TSC circuit.

Figure 6.8 shows the integration of a BIVS system and a TSC circuit. In the fault-free case, Verr=0 and Self-Test'=1 will be produced during normal operation phase while Verr=1 and Self-Test'=0 will be produced during self-exercising phase. When an indeterminate logic input is detected by the BIVS system during normal operation phase(Self-Test'=1), a non-codeword(11) will be produced. In the self-exercising phase, a non-codeword(00) will indicate possible faults within the BIVS system. A 2-pair two-rail checker is used to multiplex the error signal from the code checker and the BIVS system.

Apart from monitoring the output of the functional block, if necessary, extra BIVS can be implemented to enhance the testability of any internal node of the system. As an example, our design can be implemented to monitor the input of each cell in a standard-cell design. An additional output line of the cell may be used to detect the indeterminate logic values at its input. In this way, the overall system testability and reliability can be significantly increased.

## 6.5 SUMMARY

Indeterminate logic values present inside the functional block can contaminate the objective of implementing a self-checking system. As stated in previous sections, the detection problem is complicated in on-line testing. We have proposed a novel circuit, called built-in intermediate voltage sensor, based on CMOS technology to tackle the problem. It was demonstrated that our design can efficiently detect indeterminate logic values inside the circuit. The significant increase in testability and reliability, the benefit outweighs the penalty of the additional cost and slight performance degradation. Furthermore, our design is highly testable for transistor stuck-on and stuck-open faults. This merit makes it suitable for on-line testing applications. Since the design is based on CMOS design approach, it is suitable to be used together with Iddq testing to achieve a high system quality. In summary, Iddq testing is recognised as the most common method to detect bridging faults in CMOS digital circuits. However, unless, the speed of Iddq testing technique can be improved, our design is a good alternative solution to the problem.

# Chapter 7

# Conclusions and Future Works

## 7.1 CONCLUSIONS OF THE PRESENT WORK

Self-checking circuit design techniques can be employed to enhance the reliability of a system. With the totally self-checking(TSC) property achieved, the system can produce correct output or the system can indicate the error by means of non-codeword output under a predetermined fault model. In this chapter, a final conclusion of our contributions to the self-checking system will be given.

In Chapter 2 and Chapter 3, we reviewed the basic principles and concepts of self-checking circuits(SCCs) and carefully defined the terminology used.

Chapter 4 discusses the multiple stuck-at(s-at) fault coverage of existing self-checking checkers. Nowadays, to effectively detect the multiple stuck-at fault for a highly reliable system becomes the basic requirement. In this chapter, it was shown that the existence of multiple s-at faults could invalidate the TSC property. In some literature, it was suggested that the multiple s-at fault which escaped from codeword detection could be detected by non-codeword. However, the checker is not accessible from the primary input. Non-codeword is not available during normal operation. Based on this observation, we developed an algorithm to detect the multiple s-at fault of two-rail and parity checkers which used codewords as test vectors only. Analysis result shows that the algorithm can achieve complete detection of multiple s-at faults. The required test length is much less than the previously proposed one. Meanwhile, the algorithm can be implemented by cyclic shift register(CSR), which is based on the built-in self-test(BIST) approach easily. With the application of BIST approach, the test vectors can be easily generated at the input of the checker despite of the complexity of the functional block. Hence, a self-checking circuit which provides on-line single fault detection with off-line multiple fault detection algorithm can achieve high reliability.

In Chapter 5, we extended our study to the fault coverage problem in fully static and domino CMOS VLSI logic circuits. Open fault and bridging fault models were considered in the analysis because they were the most common physical defects in the VLSI circuits. In static CMOS logic circuits, detailed analysis showed that in the event of an open fault occurrence, a combinational circuit may behave as sequential one. However, it still produces logical error. With suitable sequence of test vectors, the open fault can be detected. However, bridging faults may include the degradation of voltage, current and timing parameters which may not result in logical error. Thus, it is hard to be detected by logic testing. In recent years, Iddq testing has been proposed to detect the bridging fault. It is based on the fact that there is no quiescent current except very little leakage current flowing during steady state. By monitoring the abnormal large supply current(Iddq), bridging fault can be detected. With the integration of Iddq testing and logic testing, it can achieve higher quality of test. However, Iddq testing is a quite expensive testing technique. A high quality current monitoring device called built-in current sensor(BICS) is required to provide precise measurement of the current. On the other hand, a precise measurement of current requires a slower measuring speed (which is relatively slower than logic monitoring). In addition, when this device is inserted in the circuit-under-test(CUT), the performance of the CUT will be degraded.

We integrated the Iddq testing with the test generation algorithm proposed in Chapter 4. The bridging fault model used consisted of the intratransistor bridging fault and gate-level bridging fault. It has been shown that the proposed test generation algorithm could also effectively detect the bridging fault under the hybrid testing scheme. Two outstanding results can be drawn. Firstly, the codeword is enough to detect all the bridging faults in the predefined, but also representative fault model. Secondly, hybrid testing scheme can achieve very high quality of test. In fact, if the operating speed of application is slow or the speed of the BICS can be increased, Iddq testing can be employed for on-line testing.

Domino-CMOS logic circuit is dynamic in nature. It has been shown that it is more testable under realistic fault model than static CMOS logic circuit. However, some bridging faults remain undetectable in the logic testing approach. In our analysis, we showed that these undetectable bridging faults were also undetectable by Iddq testing. The dynamic circuit structure inherently prohibits the application of Iddq testing technique. Furthermore, domino logic circuit requires very precise charging and discharging processes. Bridging fault may affect the magnitude of current to charge or discharge the circuit. Based on this observation, modifications of the domino-CMOS logic circuit were proposed. In the first scheme, modification was carried out by adding extra control transistor and using two clock signals to control the sequence of operation phases in the circuit. The bridging fault then became detectable. However, the control transistor itself was still undetectable. In the second scheme, the originally undetectable faulty transistor was removed from the circuit. Simulation showed that the output is not degraded. In fact, the operation speed is faster than original one. Also, rules were set up to modify the circuit structure. Result showed that this modified version was highly testable for both logic and Iddq testing schemes. In particular, the previous undetectable bridging fault problem is solved. Comparing these two schemes, the second scheme can achieve higher testability, but also requires more overhead.

Chapter 6 reports an alternative method to detect bridging fault particularly for on-line testing application. In the previous chapter, it was shown that bridging fault could be effectively detected by Iddq testing technique in CMOS VLSI logic circuit. However, the low detection speed of Iddq testing makes it hard to apply for on-line testing application. On the other hand, a major problem arises. When the output of the functional block of a self-checking system is contaminated by the bridging fault and results in indeterminate logic value, the checker may not be capable of detecting this error. And the normal operation of whole system is affected. A special logic region sensing device called built-in intermediate voltage sensor(BIVS) was developed in recent years. However, the drawbacks are: 1) the large power consumption and 2) lack of testability consideration. For self-checking application, it is important that the sensor must fulfil the self-checking properties. Also, the sensing device should be small in size, has low power consumption and minimum adverse effect on the circuit-under-test. These factors motivate us to develop a new circuit design to detect the indeterminate logic value at the faulty site. And so, the integration of BIVS system with the self-checking system was proposed. Detailed analysis showed that our design could achieve the self-checking properties. In terms of the significant increase in testability and reliability, the benefit outweighs the penalty.

In summary, the study of multiple fault coverage in self-checking circuit has been extended to different fault models including multiple s-at faults, open faults and bridging faults. In this thesis, our work has been focused on improving the detectability and testability of the fault. Overall, our work has successfully improved the dependability of SCCs.

## 7.2 FUTURE WORKS

The major problem for future research may include the design methodology for particular technology such as CMOS, by including more realistic fault model. As discussed in previous chapters, design of SCCs can be solved easily for single s-at fault model. However, it fails to achieve the TSC property for realistic fault model and has bridging fault coverage problem. In this thesis, this problem is tackled by Iddq testing and BIVS approaches. In the first approach, it is shown that the Iddq testing can effectively detect the bridging faults. However, the speed of detection is slow and the design of the current sensing device is complicated. Obviously, to design a simply and fast current sensing device for on-line applications is one of the main research area for future work. In the latter approach, BIVS system and SCC are integrated to detect the indeterminate logic at the input of the checker. However, the indeterminate logic value can still occur inside the circuit. It was shown in the past that extra test points can increase the testability of the circuit. Based on this observation, we believe that placement of BIVS can also enhance the detectability of bridging fault inside the circuit. Placement of BIVS is another main future research problem. Also, a simple BIVS is needed to reduce the cost of additional overhead. In summary, the study of realistic fault coverage problems in SCCs is important. To develop cost-effective methods to improve the fault coverage problem is the major goal in future research direction.

# References

[ABRA 90] Abramovivi, M., Breuer, M.A. and Friedman, A. D., Digital System Testing and Testable Design, Computer Science Press, New York, 1990.

[ADER 71] Anderson, D.A., "Design of self-checking digital networks using coding techniques. Coordinates. Sciences Laboratory", Report R/527, University of Illinois, Urbana, 1971.

[ADER 73] Anderson, D.A. and Metze, G., "Design of totally self-checking check circuits for m-out-of-n codes," IEEE Trans. Comput., c-22, Mar. 1973, pp. 263-269.

[BOUD 91] Boudjit, M. and Nicolaidis, M., "A tool for computation of output code spaces in complex self-checking systems", Pac. Rim Int. Symp. on FTS, 1991, pp. 122-127.

[BURN 94] Burns, S.W. and Jha, N.K., "A totally self-checking checker for parallel unordered coding scheme", IEEE Trans. Comput., 43, 1, April 1994, pp. 490-495.

[BUSA 94] Busaba, F.Y. and Lala, P.K., "Self-checking combinational circuit design for single and unidirectional multibit error", J. Electronic Testing: Theory and Applications, 5, 1994, pp. 19-28.

[CART 68] Carter, W.C. and Schmeider, P.R., "Design of dynamically checked computers", Proc. 4th Congress IFIP, 2, 1968, pp. 878-883.

[CHEE 92] Cheema, M.S. and Lala, P.K., "Totally self-checking CMOS circuit design for breaks and stuck-on faults", IEEE J. Solid-State Circuits, 27, 8, Aug. 1992, pp. 1203-1206.

[COX 88] Cox, H. and Rajski, J., "A method of fault analysis for test generation and fault diagnosis", IEEE Trans. CAD, 7, 7, July 1988, pp. 813-833.

[DIMA 95] Dimakopoulos, V.V., Sourtziotis, G., Paschalis, A. and Nikolos, D., "On TSC checkers for m-out-of-n codes", IEEE Trans. Comput., 44, 8, Aug. 1995, pp. 1055-1059.

[FAVA 90] Favalli, M., Olivo, P., Damiani, M. and Ricco, B., "Novel design for testability schemes for CMOS ICs", IEEE J. Solid-State Circuits, 25, 5, Oct. 1990, pp. 1239-1246.

[FAVA 96] Favalli, M. and Metra, C., "Sensing circuit for on-line detection of delay faults", IEEE Trans. VLSI Systems, 4, 1, Mar. 1996, pp. 130-133.

[FERG 91] Ferguson, F.J. and Larrabee, T., "Test pattern generation for realistic bridge faults in CMOS ICs", Proc. Int. Test Conf., 1991, pp. 492-498.

[FUJI 85] Fujiwara, H., Logic Testing and Design for Testability, MIT Press, Cambridge, Mass, 1985.

[FUJI 87] Fujiwara, E. and Matsuoka, K., "A self-checking generalized prediction checker and its use for built-in testing", IEEE Trans. Comput., c-36, 1, Jan 1987, pp. 86-93.

[FUJI 91] Fujiwara, E. and Yoshikawa, M., "A design method for cost-effective self-testing checker for optimal d-unidirectional error detecting codes", Pac. Rim Int. Symp. on FTS, 1991, pp. 174-179.

[GALI 80] Galiay, J., Grouzet, Y. and Vergniault, M., "Physical versus logical fault models MOS LSI circuits: impact on their testability", IEEE Trans. Comput., c-29, 6, 1980, pp. 537-541.

[GAIT 85] Gaitanis, N., "A totally self-checking error indicator", IEEE Trans. Comput., c-34, 8, Aug. 1985, pp. 758-761.

[GAIT 95] Gaitanis, N., Kostarakis, P. and Paschalis, A., "Totally self-checking reconfiguration duplication system with separable internal fault indication", Proc. 4th Asian Test Sypm., 1995, pp. 316-321.

[HIRA 91] Hirabayashi, K., "Self-checking CMOS circuits using pass-transistor logic", J. Electronic Testing: Theory and Applications, 2, 1991, pp. 205-208.

[HUGH 84] Hughes, J.L.A., McCluskey, E.J. and Lu, D.J., "Design of totally self-checking comparators with an arbitrary number of inputs", IEEE Trans. Comput., c-33, 6, June 1984, pp. 546-550.

[JACO 92] Jacob. J, "Multiple fault detection in two-level multi-output circuits", J. Electronic Testing: Theory and Applications, 3, 1992, pp. 171-173.

[JHA 84] Jha, N.K. and Abraham, J.A., "Totally self-checking MOS circuits under realistic physical failures", Proc. Int. Conf. Comp. Design, Oct. 1984, pp. 665-670.

[JHA 88] Jha, N.K., "Testing for multiple faults in domino-CMOS logic circuits", IEEE Trans. CAD, 7, 1, Jan. 1988, pp. 109-116.

[JHA 90] Jha, N.K., "Strongly fault-secure and strongly self-checking domino-CMOS implementations of totally self-checking circuits", IEEE Trans. CAD, 9, 3, March 1990, pp. 332-336.

[JHA 93] Jha, N.K., "Fault detection in CVS parity trees with application to strongly self-checking parity and two-rail checkers", IEEE Trans. Comput., 42, 2, Feb. 1993, pp. 179-189.

[JOHN 89] Johnson, B.W., Design and Analysis of Fault Tolerant Digital Systems, Addison-Wesley, Reading, MA, 1989.

[JONE 94] Jone, W.B. and Wu, C.J., "Multiple fault detection in parity checkers", IEEE Trans. Comput., 43, 1994, pp. 1096-1099.

[KARK 94] Karkouri, Y., Abouhamid, E.M., Cerny, E. and Verreault, A., "Use of fault dropping for multiple fault analysis", IEEE Trans. Comput., 43, 1, Jan. 1994, pp. 98-103.

[KAWA 96] Kawakubo, K., Tanaka, K. and Hiraishi H., "Formal verification of self-testing properties of combinational circuits", Proc. 15th Asian Test Symp., 1996, pp. 119-122.

[KHAK 84] Khakbaz, J. and McCluskey, E.J., "Self-testing embedded parity checkers", IEEE Trans. Comput., C-33, 8, Aug. 1984, pp. 753-756.

[KOND 96] Kondo, H. and Cheng, K.T., "AN efficient compact test generator for IDDQ testing", Proc. 15th Asian Test Symp., 1996, pp. 177-182.

[KUND 86] Kunda, S. and Reddy, S.M., "On the design of TSC CMOS combinational logic circuits", Proc. Int. Conf. Comp. Design, 1986, pp. 496-499.

[KUND 96] Kunda, S., Sogomonyan, E.S. and Goessel, M., "Self-checking comparator with one periodic output", IEEE Trans. Comput., 45, 3, March 1996, pp. 379-380.

[LALA 85] Lala, P.K., Fault Tolerant and Fault Testable Hardware Design, Prentice-Hall International, Englewood Cliffs, N. J., 1985.

[LIDE 92] Liden, P. and Dahlgren, P. and Torin, J., "Transistor fault coverage for self-testing CMOS checkers", Proc. Int. Test Conf., 1992, pp. 476-485.

[LEE 96a] Lee, K.J. and Tang, J.J., "Two modeling techniques for CMOS circuits to enhance test generation and fault simulation for bridging faults", Proc. 15th Asian Test Symp., 1996, pp. 165-170.

[LEE 96b] Lee, K.J., Tang, J.J., Huang, T.C. and Tsai, C.L., "Combination of automatic test pattern generation and built-in intermediate voltage sensing for detecting CMOS bridging faults", Proc. 15th Asian Test Symp., 1996, pp. 100-105.

[LEVI 81] Levi, M.W., "CMOS is most testable", Proc. Int. Test Conf., 1981, pp. 217-221.

[LO 90] Lo, J.C. and Thanawastien, S., "On the design of combinational totally self-checking 1-out-of-3 code checkers", IEEE Trans. Comput., 39, 3, March 1990, pp. 387-393.

[LO 92] Lo, J.C., Thanawastien, S., Rao, T.R.N. and Nicolaidis, M., "An SFS Berger check prediction ALU and its application to self-checking processor designs", IEEE Trans. CAD, 11, 4, April 1992, pp. 525-540.

[LO 95] Lo, J.C., Daly, J.C. and Nicolaidis, M., "A strongly code-disjoint built-in current sensor for strongly fault-secure static CMOS realization", IEEE Trans. CADICS, 14, 11, Nov. 1995, pp. 1402-1407.

[LUBA 93] Lubaszewski, M. and Courtois, B., "Reliable fail-safe systems", Proc. 2nd Asian Test Symp., Nov. 1993, pp. 32-37.

[MACI 95] Macii, E. and Wolf, T., "Multiple fault diagnosis in combinational circuits", Computers Elect. Engg., 21, 5, 1995, pp. 351-358.

[MALA 82] Malaiya, Y.K. and Su, S.Y.H., "A new fault modelling and testing technique for CMOS devices", Proc. Int. Test Conf., 1982, pp. 25-34.

[MALY 92] Maly, W. and Marek, P., "Built-in current testing", IEEE J. Solid-State Circuits, 27, 3, Mar. 1992, pp. 425-428.

[MARO 78] Marcouf, M.A. and Friedman, A.D., "Efficient design of self-checking checker for any m-out-of-n code", IEEE Trans. Comput., c-27, 6 June 1978, pp. 482-490.

[MASA 88] Shoji, M., CMOS Digital Circuit Technology, Prentice Hall , Englewood Cliffs, N.J., 1988.

[METR 94a] Metra, C., Favalli, M. and Ricco, B., "Novel 1-out-of-n CMOS checker", Electronics Letters, 30, 17, 1994, pp. 1398-1400.

[METR 94b] Metra, C., Favalli, M. and Ricoo, B., "Highly testable and compact 1-out-of-n CMOS checker", Proc. Int. Workshop on Defect and Fault tolerance in VLSI Systems, 1994, pp. 142-150.

[METR 95] Metra, C., Favalli, M., Olivo, P. and Ricco, B., "Design of CMOS checkers with improved testability of bridging and transistor stuck-on faults", J. Electronic Testing: Theory and Applications, 6, 1995, pp. 7-22.

[MEYE 75] Meyer, J.F. and Sundstrom, R.J., "On-line diagnosis of unrestricted faults", IEEE Trans. Comput., c-24, 5, May 1975, pp. 468-475.

[MILL 91] Millman, S.D. and McCluskey, E.J., "Bridging, transition, and stuck-open faults in self-testing CMOS checkers", Proc. 21st Int. Fault-Tolerant Comput., June 1991, pp. 154-161.

[MIN 88] Min, J.H. and Li, J.T., "Strongly fault secure PLA's and totally self-checking checkers", IEEE Trans. Comput., 37, 7, July 1988, pp. 863-867.

[MIUR 92] Miura, Y. and Kinoshita, K., "Circuit design for built-in current testing", Proc. Int. Test Conf., 1992, pp. 873-881.

[MOZU 96] Mozuelos, R., Pelaez, N., Martinez, M. and Bracho, S., "Built-in current sensors in mixed circuit test based on the dynamic power supply current", 2nd IEEE Int. On-Line Testing Workshop, 1996, pp. 25-28.

[NANY 87] Nanya, T. and Kawamura, T., "On error indication for totally self-checking systems", IEEE Trans. Comput., c-36, 11, Nov. 1987, pp. 1389-1392.

[NANY 88] Nanya, T., Mourad, S. and McCluskey, E.J., "Multiple stuck-at fault testability of self-testing checkers", Proc. Int. Fault-Tolerant Comput., 1988, pp. 381-386.

[NICO 86] Nicolaidis, M. and Courtiois, B., "Design of NMOS strongly fault-secure circuits using unidirectional error detecting codes", Proc. 16th IEEE Int. Symp. on Fault-Tolerant Comp., July 1986, pp. 22-27.

[NICO 88] Nicolaidis, M., and Courtois, B., "Strongly code-disjoint checkers," IEEE Trans. Comput., 37, June 1988, pp. 751-756.

[NICO 89a] Nicolaidis, M., "Self-exercising checkers for unified built-in self-test(UBIST)", IEEE Trans. CAD, 8, 3, 1989, pp. 203-218.

[NICO 89b] Nicolaidis, M., Noraz, S. and Courtois, B., "A generalized theory of fail-safe systems", Proc. Int. Fault-Tolerant Comput., 1989, pp. 398-406.

[NICO 90] Nicolaidis, M., "Efficient UBIST implementation for microprocessor sequencing parts", Proc. Int. Test Conf., 1990, pp. 316-326.

[NICO 91] Nicolaidis, M., "Shorts in self-checking circuits," J. Electronic Testing: Theory and Applications, 1, 1991, pp. 257-273.

[NICO 94] Nicolaidis, M., "Fault secure property versus strongly code disjoint checkers," IEEE Trans. CAD, 13, May 1994, pp. 651-658.

[OKLO 84] Oklobdzija, V.G. and Kovijanic, P.G., "On testability of CMOS-domino logic", Proc. 14th Int. Symp. Fault-Tolerant Comput., 1984, pp. 50-55.

[OZGU 91] Ozguner, F., "Design of totally self-checking embedded two-rail code checkers", Electronics Letters, 27, 4, 1991, pp. 382-384.

[PASC 90] Paschalis, A.M., Efstathiou, C. and Halatsis, C., "An efficient TSC 1-out-3 code checker", IEEE Trans. Comput., 39, 3, Mar. 1990, pp. 407-410.

[PIES 91] Piestrak, S.J., "Self-testing checkers for arithmetic codes with any checker base A", Pac. Rim Int. Symp. on FTS, 1991, pp. 162-167.

[PIES 95] Piestrak, S.J., Design of Self-Testing Checkers for Unidirectional Error Detecting Codes, Oficyna Wydawnicza Politechniki Wroclawskiej, Wroclaw, 1995.

[PRAD 86] Pradhan, D.K., Ed., Fault Tolerant Computing: Theory and Techniques, Vol. I and II, Prentice-Hall, Englewood Cliffs, N. J., 1986.

[RAJS 95] Rajsuman, R., Iddq Testing for CMOS VLSI, Artech House, Boston, 1995.

[RAO 93] Rao, T.R.N., Feng, G.L., Kolluru, M.S. and Lo, J.C., "Novel totally self-checking Berger code checker designs based on generalized Berger code partitioning", IEEE Trans. Comput., 42, 8, Aug. 1993, pp. 1020-1024.

[REDD 74] Reddy, S.M., "A note on self-checking checkers," IEEE Trans. Comput., Oct. 1974, pp. 1100-1102.

[REDD 94] Reddy, S.M., Pomeranz, I. and Jain, R., "On codeword testing of two-rail and parity TSC checkers", Proc. Int. Symp. on FTC, 1994, pp. 116-125.

[RODR 96] Rodriguez-Montanes, R, "Bridging defects resistance in the metal layer of a CMOS process", J. Electronic Testing: Theory and applications, 8, 1996, pp. 35-46.

[SAPO 96] Saposhnikov, V. V., Morosov, A., Saposhnikov, Vl. V. and Goseel, M., "Design of self-checking unidirectional combinational circuits with low area overhead", 2nd IEEE Int. On-Line Testing Workshop, 1996, pp. 56-67.

[SCHO 95] Schotten, C. and Meyr, H., "Test point insertion for an area efficient BIST", Proc. Int. Test Conf., 1995, pp. 515-523.

[SETH 77] Seth, S.C. and Kodandapani, K.L., "Diagnosis of faults in linear tree networks", IEEE Trans. Comput., c-26, 1, Jan. 1977, pp. 29-33.

[SHEN 88] Shen, J.P. and Ferguson, F.J., "Extraction and simulation of realistic CMOS faults using inductive fault analysis", Proc. Int. Test Conf., 1988, pp. 475-484.

[SHER 88] Sherlekar, S.D. and Subramanian, P.S., "Conditionally robust two-pattern tests and CMOS design for testability", IEEE Trans. CAD, 7, 3, Mar. 1988, pp. 325-332.

[SMIT 78] Smith, J.E., and Metze, G., "Strongly fault-secure logic networks," IEEE Trans. Comput., c-27, June 1978, pp. 491-499.

[SMIT 83] Smith, J.E. and Lam, P., "A theory of totally self-checking system design", IEEE Trans. Comput., c-32, 9, Sept. 1983, pp. 831-844.

[STOR 90] Storey, T.M. and Maly, W., "CMOS bridging fault detection", Proc. Int. Test Conf., 1990, pp. 842-851.

[STOR 91] Storey, T., Maly, W., Andrews, J. and Miske, M., "Stuck fault and current testing comparison using CMOS chip test", Proc. Int. Test Conf., 1991, pp. 311-318.

[TAKA 91] Takahashi, H., Iuchi, N. and Takamatsu, Y., "Test generation for combination circuits with multiple faults", Pac. Rim Int. Symp. on FTS, 1991, pp. 212-217.

[TANG 95] Tang, J.J., Lee, K.J. and Liu, B.D., "Built-in intermediate voltage testing for CMOS circuits", Proc. European Design and Test Conf., 1995, pp. 372-376.

[TAO 92] Tao, D.L., Hartmann, C.R.P. and Lala, P.K., "A general technique for designing totally self-checking checker for 1-out-of-N code with minimum gate delay", IEEE Trans. Comput., 41, 7, July 1992, pp. 881-886.

[TARN 95] Tarnick, S., "Controllable self-checking checkers for conditional concurrent checking", IEEE Trans. CADICS, 14, 5, May 1995, pp. 547-553.

[WADS 78] Wadsack, R.L., "Fault modelling and logic simulation of CMOS and MOS integrated circuits", The Bell System Technical Journal, 57, 5, May-June 1978, pp. 1449-1475.

[WAKE 78] Wakerly, J.F., Error Detecting Codes, Self-Checking Circuits and Applications, North-Holland, NewYork, 1978.

[WANG 94] Wang, J.Q. and Lala, P.K., "Partially strongly fault-secure and partially strongly code-disjoint 1-out-of-3 code checker", IEEE Trans Comput., 43, 10, Oct. 1994, pp. 1238-1250.

[WEST 93] Weste, N.H. and Eshraghian, K., Principles of CMOS VLSI Design: A Systems Perspective, Addison-Wesly Pub. Co., 1993.

[WEN 97] Wen, X.Q., Saluja, K.K., Kinoshita, K. and Tamamoto, H., "Transistor leakage fault location for static CMOS circuits", CAD, Test and Eval. For Dependability, 1997, pp. 297-302.