FAULT-TOLERANT HARDWARE DESIGNS AND THEIR RELIABILITY ANALYSES

A Thesis submitted for the degree of Doctor of Philosophy

by

Mahmoud Hafezparast

Department of Electrical and Electronics Engineering,

Brunel University, (The University of West London),

Uxbridge, Middlesex, UB8 3PH, U.K.

1990

Abstract

Fault-tolerance, which is a complement to fault prevention, is an effective method of achieving ultra-high reliability. By taking this approach fault-free computation can be achieved despite the presence of fault in the system. In this thesis three new fault tolerant techniques are presented and their advantages over well known fault-tolerant strategies are shown. One of these new techniques achieves higher reliability than any other similar techniques presented in the literature.

Generally fault-tolerant structures consist of four major blocks: the replicated modules, the disagreement and detection circuit, the switching circuit, and the voting mechanism. The most critical component in a fault-tolerant system is the voter because the final output of the system is computed by this component. This dissertation presents a new implementation for voters which reduces both the complexity and the occupied area on the chip.

The structures of the three techniques developed in this work are such that the complexity of their switching mechanisms grows only linearly with the number of modules but the voting mechanism complexity increases significantly. This is a better approach than those schemes in which the switching complexity increases significantly and the voter's complexity remains constant or grows linearly with the number of modules because it is easier to implement a complex voter than a complex switch (voters have more regular structures).

Extensive comparisons are made between different fault-tolerant techniques. A new reliability model is also developed for system reliability evaluation of the new designs. The result of these analyses are plotted, and the advantages of the new techniques are demonstrated. In the final part of the work an expert system is described which uses the knowledge acquired by these comparisons. This expert system is meant as a prototype of a component of a CAD tool which will act as an advisor on fault-tolerant techniques.

To the memory of my father,

to my mother ,

to my wife, and my children (Nasrin and Farzin)

CONTENTS

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Professor Gerald Musgrave for his help, encouragement and advice throughout the course of this project. My thanks also goes to my co-supervisor Dr. Robert Zimmer , for many useful discussions and invaluable advice in the final year of this project. I would like to express my gratitude to Mr. Derek Milligan, for giving me much of his valuable time, and for his constructive suggestions during the final year of this research. I would also like to thank my colleagues at Brunel University for their support and stimulating discussions.

Finally, my wife deserves very special thanks for her continued support, help, and encouragement throughout this research.

CHAPTER ONE

INTRODUCTION

1.1 - Introduction

During the 1970s and 1980s , integrated circuits (ICs) reached incredible levels of sophistication, with manufacturers fabricating circuits containing many tens of thousands of logic gates on a single chip. Fig 1.1 illustrates the remarkably rapid growth of circuit complexity during the last two decades .

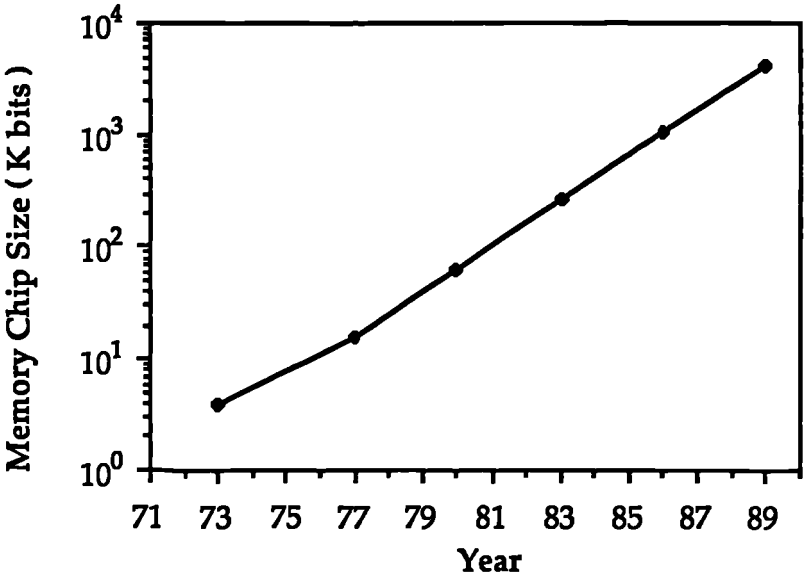


Fig. 1.1 Integrated circuit complexity versus time

This complexity has been automatically incorporated into the devices and systems implemented by these sophisticated ICs. While the number of components that can be supported on a chip is increasing, the chip itself is becoming susceptible to an increasing variety of failures, ranging from internal opens and shorts to encapsulation and bonding failures. Given the complexity of the ICs and digital systems,

and accepting that their complex design and construction are susceptible to the inherent fallibility of those who design and construct, and also taking into account the limitations imposed by the technology used, it would be surprising indeed if any modern computing system provides its intended service with ultra-high reliability. It is not sufficient just to design and manufacture complex ICs and digital systems, but system designers and manufacturers must also present measures to improve the reliability of these complex devices and systems.

In fact the drastically increased reliability requirements of digital systems forces the designers to attempt methods to achieve high reliability. As an example the reliability for the Saturn V launch computer (1964) was only 0.99 for 250 hours, in comparison to the late 1970s FTMP and SIFT computers with reliability requirements of 10^{-9} failures per hours over the 10 hour mission time.

1.2 - General Methods of Improving System Reliability

Generally there are two approaches to increasing the reliability of systems.

I - The first is the traditional approach which is called **fault prevention.** By taking this approach designers and manufacturers try to prevent system failure by ensuring that all possible causes of unreliability have been removed from the system before the system is put into service.

Fault prevention has two aspects, namely, fault avoidance, and fault removal. Fault avoidance is concerned with design methodologies and the selection techniques which aim to avoid the introduction of faults during the design and manufacturing of a system. The use of reliable components is an example of fault avoidance. Fault removal is concerned with checking the implementation of a system and removing those faults which are exposed .

Design For Testability (DFT) which concerns the improvement of the controllability and the observability of VLSI circuits to ease testing of these devices has been very successful , but even under this technique testing of VLSI devices is a serious problem for the designers and manufacturers of these devices. In many ways, testing a very large scale integrated circuit is more difficult than designing it. It is both possible and likely that a large integrated circuit will contain embedded elements that cannot practically be tested even when methods such as exhaustive testing (i.e. testing every conceivable operating state) is employed .

For complex circuits, exhaustive testing becomes unrealistic. For instance, an exhaustive test of the 8080 microcomputer, only modestly complicated by today's standards, would take over 10 to 20 years, at one million tests a second [Fe 83], or a microprocessor such as Motorola 68000 would take many years of CPU time to test exhaustively. Thus one may conclude that the primary stumbling blocks in VLSI circuit development are therefore testing of the devices, not design and fabrication. The problem of testing VLSI devices is aggravated by the

shortage of test engineers and the high costs of testing, in addition to the difficulties of developing programmes that control the Automated Test Equipment (ATE) .

However, despite the adoption of fault prevention techniques, *faults will occur during the operation of systems* . So when operation without failure is required despite the presence of faults, the adoption of the above strategies *alone* in general is insufficient. There is also an *upper limit* for improvement of component or system reliability due to design methodology, cost limitations , and available manufacturing techniques. Indeed this is the most important reason behind the implementation of designs taking another approach called *fault-tolerance* .

II - The second approach for increasing the reliability of systems is **fault-tolerance** . By definition a fault tolerant computing system is a system which can compute correctly even with the presence of faults in its hardware or its software. The important objective of fault-tolerant design is to enhance the reliability of digital systems which can not be achieved by adopting the fault-prevention approach. Apart from ultra-high reliability needs, fault-tolerant computing is driven by other key factors, such as *ultrahigh availability* (e.g. the ESS system of Bell Telephone, which has an availability requirement of only 2 minutes down-time per year [To78]), *reduced life-cycle costs* (which is a major manufacturing objective in commercial computers), *and long-life*

applications (for instance, the very high survival problem warranted in spacecraft computers such as the one planned for the Galileo spacecraft).

Other major factors influencing growth and development of fault-tolerant computers include the tradeoffs between the lack of high reliability and the loss of computational power. Also the use of computers at the *critical points* makes it essential that they not only be *highly available* , but *especially , reliable* , so as to encourage their acceptance and use by the general public. A good example here is the ultimate goal of paperless offices and banks which is impossible to achieve without the availability of low-cost, highly available, and highly reliable computers.

The design of highly reliable computers is actually much more complex than the design of other complex human made objects (e.g. robots, airplanes, etc.). Perhaps this can be better grasped by looking at one statement in the IEEE Spectrum (Oct 81 p.41): "Information processing errors can occur through a failure lasting a billionth of a second in one of the hundreds of thousands of digital components that switch billions of times a day,".

1.3 - Fault-tolerance in systems

Fault-tolerance is incorporated in a system by adding redundancy (i.e. a system or a component will be replicated many times). The redundancy

can be in the form of software, hardware, or a combination of both.

To obtain the correct output of a system designed to tolerate failures, the following blocks are generally used.

I) A voting mechanism to vote on the outputs of the replicated modules or components.

II) A disagreement detection circuit to detect any failures occurring during the operation of the system.

III) A switching mechanism to take measures for reconfiguration of the system when failures occur.

This research discusses the above compartments in detail and develops three new fault-tolerant designs to improve the overall system reliability

The first design concentrates on the number of gates used in the switching mechanism. As a result a switching circuit is developed which use fewer gates than other similar designs proposed by others.

In the second design, the disagreement detection circuit will be optimised as well as the switching mechanism. The switch structure in this design is such that it does not propagate the failures from one component to the other switch components. This feature has a beneficial effect for reliability improvement.

The structure of the switch in the third design is such that it has the same features as in the second design, in addition, it can tolerate more failures than other techniques including the above schemes, thus a better reliability improvement can be achieved by this technique.

As the voting component in any fault-tolerant design is the most critical component, an approach will be presented in this work to implement this component as simply as possible. To be able to implement a highly reliable voter, a modular structure is used to minimise the chip area as well as using as few transistors as possible.
A new reliability model has been developed and used in an extensive comparison of fault-tolerant techniques (including the new designs) . The reliability improvement made by the designs is also shown.
The last part of this research involves the initial development of an expert system which can be used as part of a CAD tool. The expert system will use the knowledge resulting from the comparitive study to advise on the fault-tolerant technique that best suits a particular application.

A short detailed break-down of the dissertation is given below:
Chapter Two reviews the basics of reliability theory definitions and a few useful definitions used throughout this work.
Chapter three focuses on the general methods that can be used to achieve highly reliable computing system which is the objective of this research. After a short discussion about fault-prevention approach, the complementary approach, that is fault tolerance will be discussed and a description of a few fault-tolerant designs will be presented for the purpose of reliability comparison.
Chapter four discusses a powerful fault-tolerant technique published recently and shows that the published technique is not efficient with a

large number of modules. An improvement to this technique will be proposed which will operate correctly and requires fewer gates. Then two new fault-tolerant techniques will be presented. It will be shown that reliability improvement and higher fault-tolerance can be achieved by the adoption of these designs.

Chapter five discusses the classical reliability model and compare the reliability of triple modular redundancy systems (TMR), N-tuple Modular Redundancy (NMR) system, and hybrid redundancy techniques with the reliability of a non-redundant system. Then a new reliability model will be presented and the reliability of systems using the new fault tolerant techniques mentioned in chapter four will be evaluated by the new model. Finally a new approach will be discussed for the implementation of the voters used in all fault-tolerant designs. This design approach for the voters dramatically reduces the number of transistors for their implementation, particularly when the number of basic modules in the fault-tolerant designs exceeds three. Consequently the overall reliability of the system will be improved.

Chapter 6 deals with an expert system (that could be part of a CAD system) on fault-tolerant techniques and their reliabilities and will operate as an adviser.

Chapter seven presents the conclusion and future research that can be done in this area.

CHAPTER TWO

RELIABILITY

2.1 - Introduction

Digital systems undertake a great variety of important tasks such as controlling nuclear power stations, monitoring patients in hospitals , space programmes, agriculture and production lines in large and small plants. Considering these important applications , the reliability of digital systems should receive more attention. Thus more effort must be made to study, design, and evaluate reliable systems. Generally a reliability engineer is concerned with a wide rang of topics, which make his task more difficult. Apart from the environmental conditions and specifications, the reliability engineer should be involved in mathematical aspects such as probability and statistics, some physics in the study of failure, and electronics for components and product characteristics. Therefore a reliability engineer faces a wide variety of physical and mathematical problems in addition to those arising from his own area of reliability engineering.

To design and evaluate reliable systems, it is helpful to review some previous work in this area and investigate the techniques which have been used for reliability improvement. But first, study of reliability principles and some basic definitions as the mathematical background for reliability analysis and modelling is necessary. Understanding these principles and methods is an essential ingredient of the analysis.

2.2 - Reliability and Cost

Cost is an important parameter in the design and manufacture of a digital system. Users of any system are usually aware of the extra cost involved with imperfect and unreliable systems. Manufacturers of domestic products such as T.V.s, washing machines, and similar products, suffer high costs due to failures under warranty. The cost of system down-time and unreliability in communications, telephone switching, airlines, military and public services is often very high. This is however on top of the extra costs due to the system failure.

In order to have an idea of the total cost, it is useful to divide it into different groups .

First: the initial purchasing cost including design, development and manufacturing.

Second: a- the maintenance and repair cost during system operation.

b- the cost of standby equipment for reliability improvement.

Attempts to increase reliability, rapidly increase the design and development costs, and therefore the initial purchase costs. On the other hand the maintenance is less for more reliable systems.

The total cost of a system (known as cost of ownership) [PaOco 81] and its relation with initial purchase, and other costs is shown in Fig.2.1. However there is a point that reliability can not be further improved (either economically or practically) , as shown in the graph.

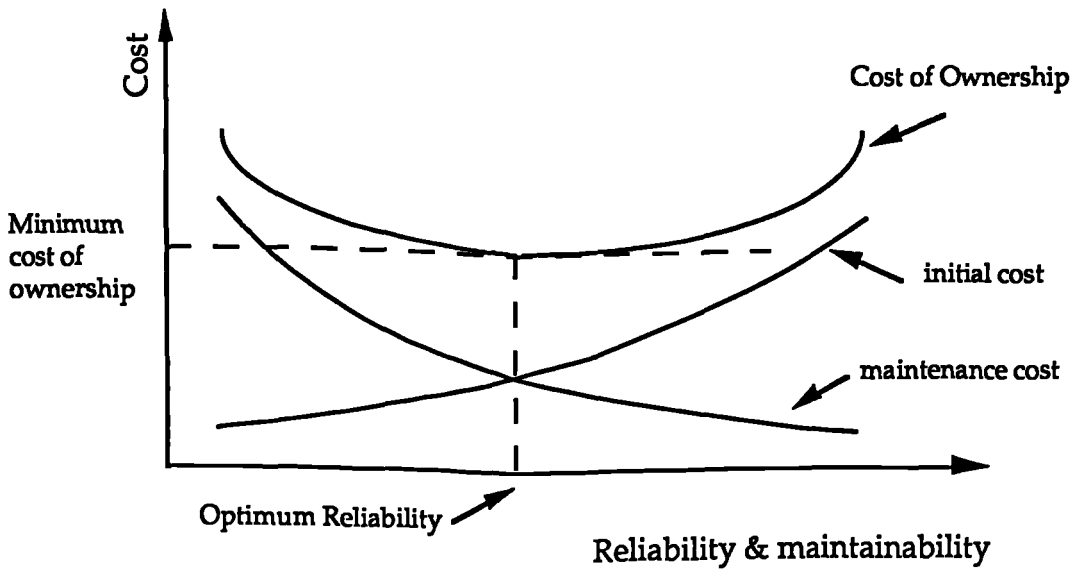


Fig 2.1 - Reliability agaist cost

2.3 - Reliability Definition

There is no dispute about the need for reliable systems, but some vague notion of reliability is not sufficient in engineering. Exact definition and additional quantitative value is needed. Reliability of a digital system can be defined as follows:

> The reliability of system (measured at a time T) is the probability that the system has not failed up to time T, subject to specified environmental conditions, e.g. specified temperature, vibration, humidity and so on [Fe 57].

Thus a value can be assigned to the reliability of a system. For example

a piece of equipment can have a reliability of 95 per cent over a 400 hour period, subject to a maximum vibration of V and the temperature in the range of 15 ^{0}c to 30 ^{0}c, and a mild humidity. The above definition of reliability is generally accepted, but this definition is not a complete definition for the whole life of a system from the starting time to the end. In practice, however more details about a system are needed. Thus more factors should be considered. For instance, the age of a digital system is a factor that should be taken into account. For a correct operation in a specified period under a specified condition, the system must be operating correctly at the beginning of the observation period . But the definition does not distinguish between a new system , which is starting its life, an established system which has been operational for a considerable time with its faults already corrected and a system which is been used for a long time and due wear out. For a new equipment, generally there is an initial period of high failure rate , which takes some times before the faults are detected , located , and repaired . During this period the failure rate falls rapidly to a value which is almost constant over a long period. After the initial period the useful life of the system starts. The reverse situation applies with regards to system wearing out , since the failure rate increases sharply as the system gets older. Fig. 2.2 shows the failure rate in these periods. The shape is often referred to as a bathtub curve [MuIaOk87]. The above reliability definition may be quoted for the useful life period.

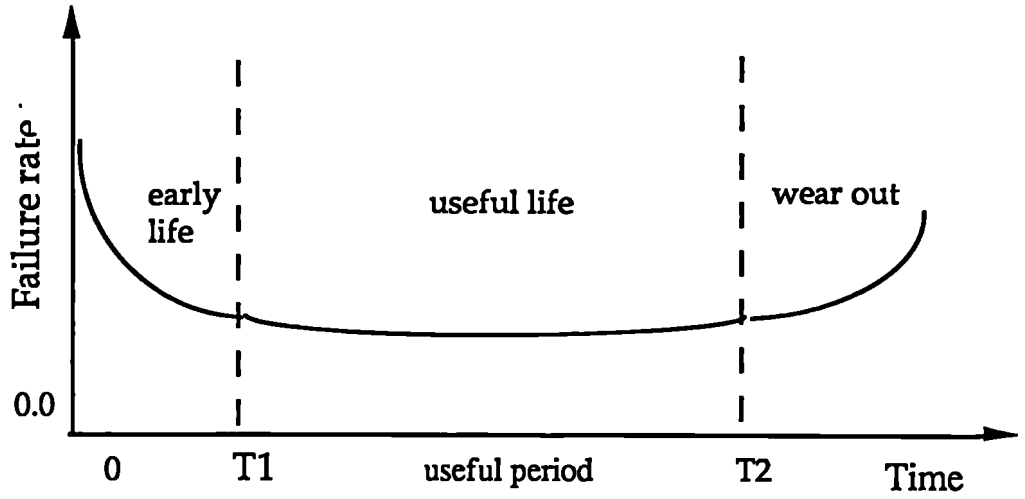


Fig 2.2 Failure rate as a funtion of time

The graph shows the failure rate as a function of time. There are some basic definitions in reliability that should be reviewed before any discussion about the reliability estimation of a system.

2.4 - Failure Rate

The failure rate is defined as the number of failures per unit time, compared with the number of surviving components. Often this is assumed to be constant during the useful life of the system and is represented by λ.

2.5 - Relation between reliability and failure rate

Suppose that a system consists of N identical components. Let S(t) be the number of surviving components at time t (i.e. the number of components still operating at time t), and Q(t) is the number of components that failed up to time t. Then the probability of survival of the components also known as the reliability R(t), is given by

$$2.1 \qquad R(t) = \frac{S(t)}{N}$$

The probability of failure of the components up to time t is given by

$$2.2 \qquad F(t) = \frac{Q(t)}{N}$$

Since $S(t) + Q(t) = N$ we must have

$$2.3 \qquad R(t) + F(t) = 1 \qquad \text{or} \qquad F(t) = 1 - R(t)$$

An important function derived from F (t) is its derivative with respect to time, which often will be used in reliability analysis.
Since F (t) is a probability, its derivative is a probability distribution function, and is defined as

$$2.4 \qquad f(t) = \frac{dF(t)}{dt}$$

f (t) shows the probability of failures per unit time. Using equation 2.3

$$2.5 \qquad f(t) = \frac{d[1 - R(t)]}{dt} = \frac{-d\,R(t)}{dt}$$

Hence the probability of a failure during the period from 0 to time t is

$$2.6 \qquad F(t) = \int_0^t f(t)\,dt$$

Now the failure rate (λ), as defined earlier, can be derived as follows:

$$\text{Failure rate} = \frac{\text{The number of failure per unit time}}{\text{The number of surviving components}} \qquad \text{or}$$

$$\lambda = \frac{1}{S(t)} \cdot \frac{d\,Q(t)}{dt}, \text{ using equations 2.1 and 2.2} \qquad \lambda = \frac{1}{N \cdot R(t)} \cdot \frac{N \cdot d\,F(t)}{dt}$$

$$2.7 \qquad \lambda = \frac{1}{R(t)} \cdot \frac{d\,F(t)}{dt}$$

Using equation 2.3 the failure rate can be written as:

$$2.8 \qquad \lambda = \frac{-1}{R(t)} \cdot \frac{d\,R(t)}{dt}$$

This expression may be integrated from 0 to time t, giving

$$\int_0^t \lambda\,dt = -\int_1^{R(t)} \frac{d\,R(t)}{R(t)}$$

The limits of integration are obtained as follows:

at time t = 0, R(t) = 1 and at time t by definition the reliability is R(t).

Assuming λ is constant, we obtain

$$\lambda t = -\log R(t) \qquad -\lambda t = \log R(t)$$

$$2.9 \qquad R(t) = \exp(-\lambda t)$$

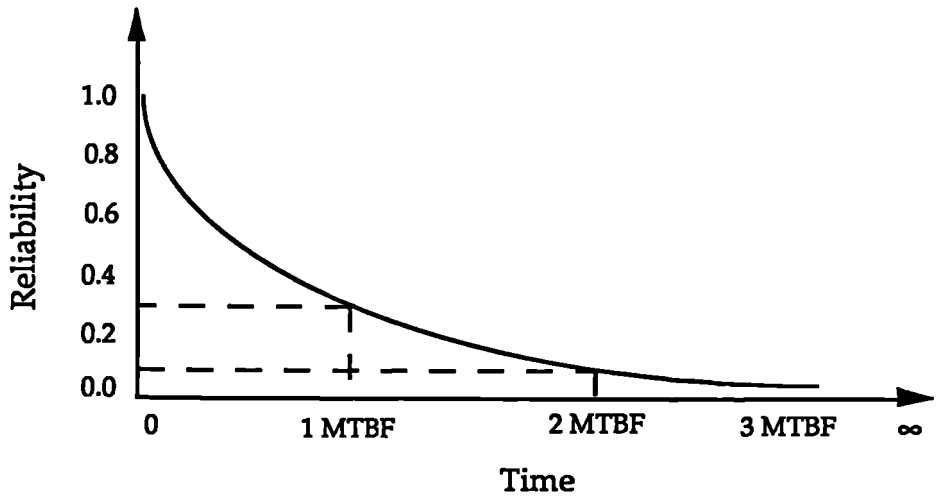


Fig 2.3 Reliability as a function of time

Therefore system reliability can be computed using the above equation, assuming that the failure rate λ is constant. Note that equation 2.9 can be considered as a general expression for reliability evaluation. The reliability figure obtained by the above equation is not the ideal for practical use, because the system reliability will be different for different time periods. For this reason another reliability measure is used which is not depended on different operating time.

2.6 - Mean Time Between Failure (MTBF)

A useful comparison of the reliability of different systems is the, Mean Time Between Failure (MTBF), which is the average time that a system will run between failures. If f(t) is the probability of failure per unit time, then MTBF can be expressed by equation 2.10.

$$2.10 \qquad MTBF = \int_0^\infty t\, f(t)\, dt$$

Using equation 2.5

$$MTBF = -\int_0^\infty t \cdot \frac{dR(t)}{dt} \cdot dt$$ and integrating by parts we obtain

$$MTBF = -[t \cdot R(t)]_0^\infty + \int_0^\infty R(t)\, dt$$

At t = 0, R(t) = 1, so t . R(t) = 0. As t increases R(t) decreases and as t tends to ∞, t . R(t) tends to zero. Thus the first term in the above equation is zero, and the general expression for MTBF, with λ as a function of time, will be given by

$$MTBF = \int_0^\infty R(t)\, dt$$

The above equation can be used for any failure distribution. Under the

assumption of constant failure rate, MTBF is given by

$$2.11 \qquad MTBF = \int_0^{\infty} \exp(-\lambda t)\, dt$$

$$2.12 \qquad MTBF = -\frac{1}{\lambda}\left[\exp(-\lambda t)\, dt\right]_0^{\infty} = \frac{1}{\lambda}$$

Therefore the MTBF of a system is the reciprocal of the failure rate. If λ is the number of failures per hour, then the MTBF is expressed in hours. The MTBF as defined above, is a concept applicable to any type of equipment which can be repaired by the replacement of a faulty component or unit. However, if all other parameters are the same, then the equipment with the greatest MTBF will be the most reliable, regardless of the period of observation. Thus MTBF provides the most convenient way of reliability comparison. The difficulty with this approach is the time needed to repair a fault. If this is the same in all cases then the equipment with the greatest MTBF will be preferred. However, there may be circumstances in which a short repair time is more important than a long MTBF, so other measures of reliability are needed.

2.7 - The Mean Time To Failure (MTTF)

The MTBF is a measure of reliability for repairable equipment. A similar measure is useful for components that can not be repaired or are more cheaply replaced. The correct measure for these components is the Mean Time To Failure (MTTF). This may be calculated from the results of life testing as follows. Let a set of N items be tested until all have failed, the time to failure being $t_1, t_2, ..., t_n$. Then the observed MTTF is given by

$$2.13 \qquad M = \frac{1}{n}\sum_{i=1}^{n} t_i$$

The failure rate will as before be given by

$$2.14 \qquad \lambda = \frac{1}{M}$$

if λ is independent of time.

2.8 - Availability

Sometimes the users of digital systems are concerned with other factors as well as the reliability of the systems. The reliability tells them the probability of system operation in a certain period with a certain

condition without a failure. Although it is valuable, the users need further information which takes into account the time lost due to repairing faults. In other words the user need to know what is the available period of the system for useful work. If we represent U as the up-time or the available period of the system and D as the down-time, availability can be defined as

$$2.15 \qquad A = \frac{U}{U + D}$$

$$2.16 \qquad A = \frac{U}{U + (\text{number of failure} * \text{MTTR})}$$

where MTTR is Mean Time To Repair, and defined as the average time needed for a failed system to be repaired and restored to working order.

$$A = \frac{U}{U + (U * \lambda * \text{MTTR})} = \frac{1}{1 + (\lambda * \text{MTTR})}$$

$$= \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}} \qquad \text{since } \lambda = \frac{1}{\text{MTBF}}$$

Now if we reduce MTTR, availability will be increased and the system will be more economical. A system where faults are rapidly diagnosed is more desirable than a system which has a lower failure rate but difficult to detect and locate the fault, and consequently longer down-time is needed for repair.

2.9 - Reliability Prediction

To predict the reliability of a complex system we partition the system into subsystems or the components used to construct it. Then the assessment of system reliability can be constructed from the probability theory for these systems. The subsystems or components are connected either in series or in parallel or both. Therefore to illustrate the functional relation between the various components of the system and the way in which a failure of each component would affect the overall system performance, we use a reliability model.

We consider three models.

1) Series system

In this model the components are connected in series. To have an operational system, all of the components should be operational and to have a correct output, all of them must work correctly i.e. a failure in any of the subsystems causes system failure. Fig. 2.4 shows the structure of this model.

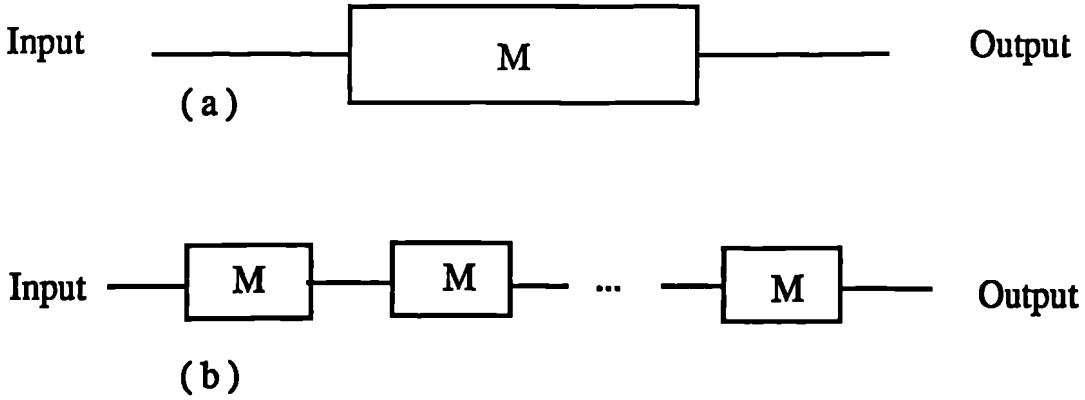


Fig 2.4 (a) - a basic module
(b) - replication of the basic module in series

In this arrangement if the reliability of each subsystem is R_i , the overall system reliability is

$$2.17 \qquad R = R1 * R2 * \ldots * Rn \qquad R = \prod_{i=1}^{n} R_i$$

2) Parallel systems

In the previous model we had no redundancy. In other words , to achieve the correct operation, the presence and correct performance of each component for construction of the system was necessary and essential. For a minimum design and production cost to carry out a specific task, series system is the normal choice of the designer. In parallel system we use extra or redundant components which are not necessary for performance of a specific task, but to increase the reliability of the system. The general principle requires the provision of more than one way of meeting the functional requirements of the system. Therefore if one of the components fail, it can not affect the system's output, while other subsystems or components are operating satisfactorily. Such a system is called a redundant system. In this model the designer will be able to improve the reliability, but there is a price to pay for this reliability improvement, that is the design cost and the manufacturing cost will be increased. Fig 2.5 shows a system using this model. In any digital system using this structure, there is a switching mechanism at point B shown in figure 2.5 to select and switch out the output of the system. There are different arrangements of this structure

which will be discussed in the later chapters.

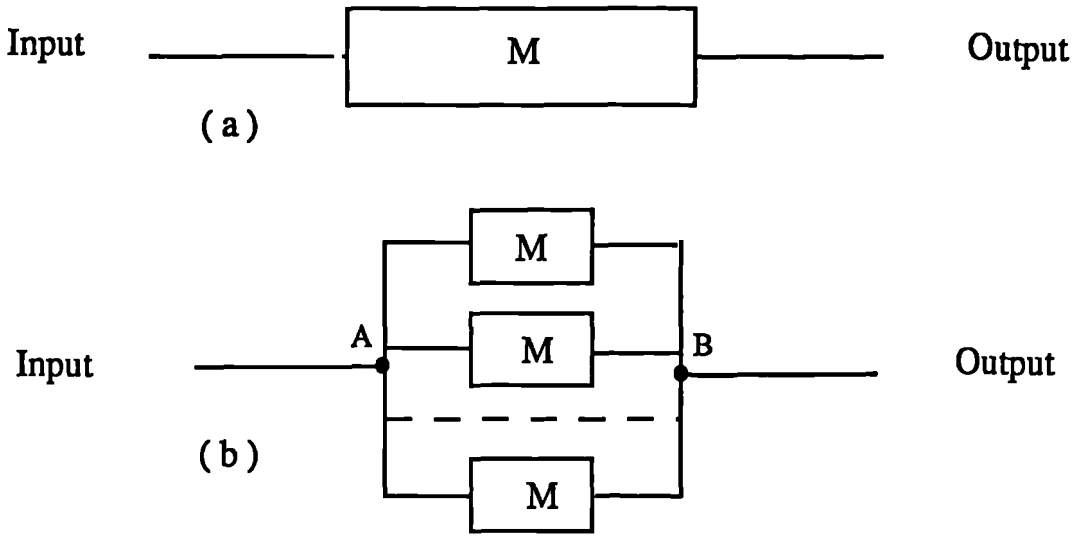


Fig 2.5 (a) - a basic module
(b) - replication of the basic module in parallel

If only one of the modules is active and others used as spares, then the overall system reliability of such a system will be obtained by equation 2.18 (provided that there is a mechanism to check the operation of the active module and reports if it fails).

$$R_{sys} = 1 - (1 - R_m)^N \qquad 2.18$$

where R_m is the reliability of the original system, R_{sys} is the overall reliability of the system. In the above equation, the switching mechanism is assumed to be fault free, and the reliability of all modules assumed to be equal.

An example of using this structure is the Bell Electronic Switching System (ESS) [To 78].

In practice a system normally consists of a combination of series and parallel subsystems. Fig 2.6 shows mixed interconnections of a few subsystems.

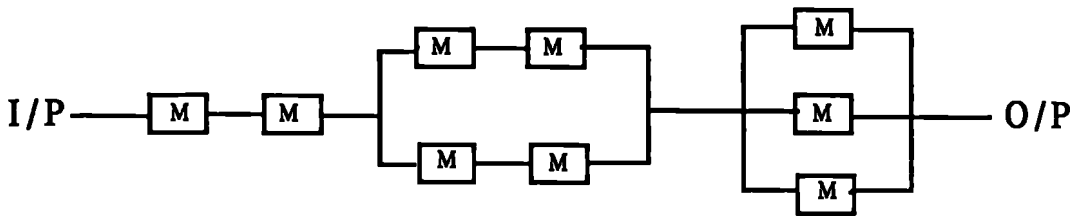


Fig 2.6 Mixed arrangement of series and parallel

The mixed arrangement is frequently used where some part of the system is particularly prone to failure and is consequently duplicated or triplicated. Examples are Pluribus system [KaEt 78], Sperry Univac 1100/60 computer [BoLiSe 80], and Computer Voter Multiprocessor (C.vmp) [SiEt 78].

2.10 - Summary

In this chapter the mathematical background necessary for reliability analyses, was developed.

The relationship between cost and reliability was shown by a curve known as cost of ownership. The curve reflects that more reliable systems have lower maintenance costs but are more expensive to buy. This curve can be used as a guide for customers.

The basic definitions for reliability evaluations were reviewed. It was shown that the exponential distribution is the most suitable distribution to be used for reliability analysis of digital systems because the failure rate is approximately constant during the useful life time of systems (Fig 2.2).

We also showed how to compute the reliability of a system from the reliability of its subsystems (modelled in series, parallel, or combination of both) .

CHAPTER THREE

METHODS OF IMPROVING SYSTEM RELIABILITY

3.1 - Introduction

As we mentioned in the previous chapters, the rapidly increasing application of computers to areas where the loss of real-time computing power could be catastrophic has brought with it the need for very high reliability. For example process control systems in big plants, control systems in nuclear power stations, or systems which undertake patients' monitoring in care units and the like, should be operational at all times, and must operate continuously without interruption . This means that a failure must be diagnosed , and appropriate measures should be taken to repair or reconfigure the system within a fraction of a second. Therefore techniques should be designed , developed , and applied to minimise or even eliminate service interruptions of the system. In another words appropriate techniques should be used to increase the reliability of the system.

There are generally two approaches to the improvement of reliability of computing systems. The first approach is called fault-prevention (fault intolerance) , and the second one is fault-tolerance. In the next section we briefly describe the fault avoidance approach . Then in the following sections the fault-tolerance approach will be reviewed.

3.2 - Fault - prevention

The objective of the fault-prevention approach is to construct systems so as to reduce the possibility of a failure by, for example using high reliability components, or adding circuitry that make it easier to test the system (design for testability). In addition, a design rule that limits the fan-out of gates, will decrease power dissipation and therefore reduce thermal effects , thus reducing the probability of hard failures. Fan-out limitation also increases the effective noise margin at the inputs of subsequent gates and thus decreases the possibility of a transient fault. Human errors can be minimised by measures such as labelling , documentation , and producing components and boards that can only be used or assembled in the correct way .

In practice however it is impossible to design and develop a system in which there is a guarantee that no failure will occur. During its manufacturing and operation time , a component or element may fail which may cause the entire system to fail (hence the name fault intolerance).

There are many cases in which fault prevention alone cannot meet system design specifications. In these cases fault-tolerance techniques should be used. In the fault tolerance approach, faults are expected to occur during the operation of the system, but by using redundancy, the faults will be masked or the faulty units will be replaced by good units automatically (reconfiguration) without any interruption in the system

operation: the system can continue to function correctly in spite of fault presence.

3.3 - Fault - Tolerance

Fault-tolerance is defined as the ability to produce correct results even in the presence of faults, [Av67] [GoLeSh66] [GrMiRo62]. Fault-tolerance is not a replacement of fault-prevention approach but a complement to it. Research activities in the area of fault-tolerant design has increased recently due to the following factors which have had a major impact on the design of these systems.

I- Advances in Very Large Scale Integration (VLSI) technology have resulted in complex chips. This complexity could make the IC's susceptible to a diverse variety of failures and could lead to a decrease in reliability.

II- Lower cost of extremely complex components and devices have made it economical to introduce redundancy into the system.

III- Testing of complex components and systems are time-consuming and expensive, moreover there is a shortage of test equipment and experts.

IV- There is an ever-increasing demand for high reliability systems to undertake safety-critical applications, despite the fact that there is an upper limit to the reliability levels that can be achieved, using the fault-prevention approach.

V- In many applications, the system down-time needs to be minimised or even eliminated to improve the availability of the system.

Basic to the design and implementation of fault-tolerant computing systems are consideration of the following three factors.

Firstly, it is necessary to identify the basic principles which underlie all fault tolerant systems. Principles that can be applied at all levels in a system.

Secondly, the measures and mechanisms to support and implement techniques based on these principles must be investigated.

Thirdly, a framework is required to support a well structured approach to fault-tolerance in order to ensure that the additional complexity introduced by the fault tolerance techniques does not reduce rather than increase the reliability of the system .

3.4 - Principles of Fault - Tolerance

To prevent faults leading to system failures five phases should be identified.

I) Error detection

The presence of a fault in a system can produce an error which can cause a failure in the system. In order to tolerate a fault in a system generally its effects must first be detected, therefore an error detection mechanism should be deployed.

II) Reconfiguration

If a fault is detected and a permanent failure located, the system should be able to reconfigure its components to replace the failed component or to isolate it from the rest of the system.

III) Retry

In many cases a second attempt at an operation may be successful. This is particularly true in case of a transient fault. Thus a retry mechanism should be available in the system to handle these cases.

IV) Reset

An error may cause too much damage to the system such that retry can not be successful and recovery may not be possible. In this case the system needs to be reset or restarted and therefore the design should provide these facilities.

IV) Fault treatment and continued service

After detection and (if necessary) reconfiguration, the effects of errors must be eliminated, and retry or reset should be used to check if the component can be used again (e.g. if failure caused by transient error). If possible the failed component should be replaced, repaired, and then put back to service.

It is possible to have a fault-tolerant design with different permutations of this procedure. However to have an effective independent system all five phases are required.

Having identified the principles of fault-tolerant design, now their implementation in hardware systems must be considered. The question of *how* fault-tolerance can be implemented in a system will be addressed in the next sections. But the questions of *where* fault-tolerance is actually required and *how much* is necessary, which concerns the *reliability requirement* , will be discussed in chapter five.

3.5 - How fault-tolerance can be implemented

Fault-tolerance can be achieved by incorporating redundancy into the system, [Kn63] [Kn64]. This redundancy can be in the form of extra hardware [SuDuCa80] [Ha89] , extra software[ChAv78], or a combination of both [AvEt71].

Software redundancy can be divided into two parts :

I- Redundancy in space, such as Error Correcting Code (ECC).

II- Redundancy in time such as repetition of some part of the software.

Hardware redundancy on the other hand is the repetition of a component or a module plus a mechanism to either mask a fault

[Ne56] or detect the faulty module and switch it out and replace it with another good module if one is available [MaAv70]. In this thesis we are dealing with hardware redundancy techniques.

3.5.1 - Hardware Redundancy

One method of increasing reliability is to introduce redundancy into the circuits to design fault-tolerant systems,[CaJeBo70] [Fl58] [Ha89]. One important point to be noted by a designer is that fault-tolerance is not a replacement to the principles of reliable system designs, but rather a supplement to them . The most important of these principles that should be remembered are as follows:

(a) Use the most reliable components

(b) Keep the system as simple as possible

3.6 - Classification of Redundancy Techniques

There are four major hardware redundancy techniques [Sh68] which will be explained, the system reliability will be calculated for each of these techniques, and finally the advantages and disadvantages of each technique will be discussed. These techniques are:

REDUNDANCY (hardware)

I- Static or masking redundancy

II- Dynamic or standby sparing

III- Hybrid

IV- Responsive Redundancy techniques

3.6.1 - Statistic Redundancy (masking redundancy)

In this technique the effect of errors is masked by the use of a voter in hardware. The most common masking redundancy is called. Triple Modular Redundancy (TMR) [Ne56]. In this technique any module or component will be repeated three times and there will be a voting scheme on the outputs of the three modules: the system output is the majority output of the modules. As long as no more than one of the modules fails, the output of the voter will be the same as the outputs of the other two fault free modules, but after the occurrence of the first failure the system is in a degraded state. The voter could be a majority voting circuit or a threshold circuit. It is quite useful to partition an arbitrary TMR network into independent cells, so that a failure in one cell can not combine with a failure in another cell to cause system failure. Fig. 3.1 shows the TMR structure.

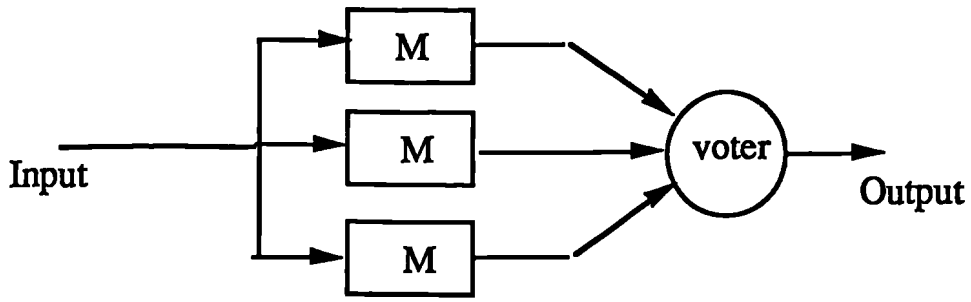


Fig 3.1 Triple Modular Redundancy (TMR) structure

3.6.2 - Cost to be paid for TMR (disadvantages)

1_ Area over head, which could be many times of the area used by the original system , particularly if the replicated component has many outputs, and if voting mechanism is deployed more than once.

2_ Power over head, at least 3 times, because all modules should be powered plus the voter. Again this could be much higher than 3 times.

3_ Increase in cost, (but using cheaper chips partially reduces the overall cost of a TMR system and make it more practical and economical).

4_ The voter is critical component: failure in the voting circuit causes system failure. (This may be partially overcome by triplicating the voter until the final stage of the system.)

5_ Failure in any module is not revealed to the operator unless extra circuitry is added to do so. Without this extra circuitry, after the first error in one of the modules, the system is less reliable than the non-

redundant system. It is possible to add extra simple circuitry to reveal the module failure and switch to simplex TMR and or repair the faulty module if it is repairable. In the long term, the system reliability is less than that of a non-redundant system.

3.6.3 - Advantages of TMR

1_ It increase the system reliability in short term missions

2_ It can tolerate one failure without delay

3_ It can tolerate all the transient errors without delay as long as they do not overlap.

4_ It does not need a complex design for the voter or for the TMR system itself. Therefore, it is recommended for short term mission and highly reliable systems.

General form of TMR in which the system contains N identical modules instead of three, is called N Modular Redundancy (NMR). Generally N is an odd number. Increasing N does not mean increasing reliability. In fact for N>7 the voter complexity goes up and causes the voter to be less reliable. But the value of N depends on the system that Multiple Modular Redundancy is applied to, i.e. for some system N=7 is also too high.

3.6.4 - Dynamic Redundancy Technique (standby sparing)

In contrast to the static technique, dynamic redundancy provides error detection capability [Sh68] [Fl58] within system. This technique uses only one active module and several spare modules. Switching circuits are needed for detecting and switching out the faulty active module, and switching-in, a good spare module (reconfiguration) [CaSc68]. So in this technique terminal activity plays an essential role (involving fault detection, diagnosis and the resultant reconfiguration). Fig 3.2 illustrates the concept of dynamic redundancy.

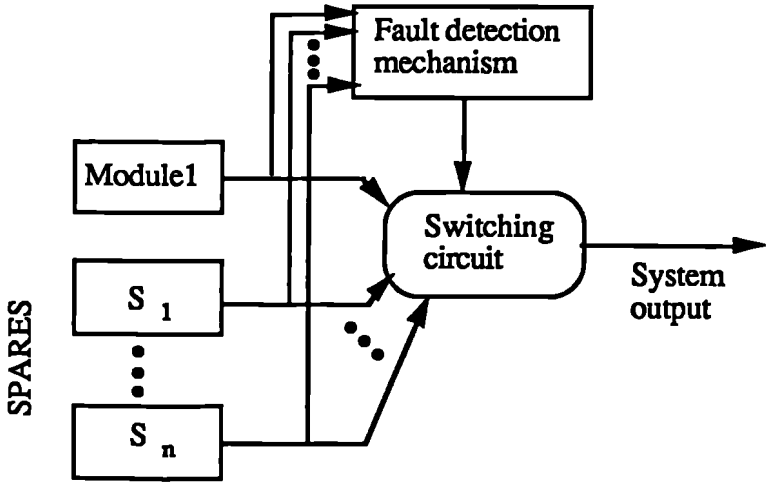


Fig . 3.2 A dynamic Structure with one active module and several spares

In general, a dynamic redundant system can be divided into two categories :

I - Cold-standby system

II- Hot standby system

In a cold standby system one module is powered up and operational, the other modules are not powered. Replacement of a faulty module by a spare is effected by turning its power off and powering a spare. In a hot standby system all the modules are powered up and operating simultaneously. If the outputs of all modules are the same, the output of any arbitrarily selected module can be taken as the system output. When a fault is detected in a module the system is reconfigured, so that the system output comes from one of the remaining fault free modules. The detection of a fault in an individual modules of a dynamic system can be achieved by using one of the following techniques:

1 _ Periodic tests

2 _ Self checking circuits

3 _ Watch_dog timers

In periodic tests the normal operation of the functional module is temporally suspended and a test routine is run to determine if faults are present in the module. A disadvantage of this technique is that it can not detect temporary faults unless they occur while the module is tested.

Self checking circuits provide a very cost effective method of fault detection. They are designed so that for normal circuit inputs they provide correct output or indicate the presence of a fault in a module. Self checking designs are very popular and there are many different systems which employ self checking circuits [Av67] [AvGiMa71] .

Watch-dog timers are an effective and popular method of fault detection. Their principle of operation is relatively simple. Timers are set to certain values at preestablished points, called checkpoints, in the programme executed by a module. A timer at a particular checkpoint counts down while the module performs its function, and is normally reset before the next checkpoint is reached. However, a software bug or a hardware fault will prevent the programme from resetting the timer. The timer then issues an interrupt command which causes automatic switch over to a spare module. The Pluribus system [KaEt78] makes substantial use of watch-dog timers. The console processor of VAX 11/780 also uses a watch-dog timers. After the detection of a fault in one of the modules it should be switched out, but one should determine whether the fault is transient or permanent, otherwise a good module may be removed because of a temporary fault. Therefore the retry technique should be used to prevent any fault-free module being switched out.

3.6.5 - Some examples of dynamic redundancy techniques.

An example of using only one spare is duplex system. In this system there are two modules in parallel with either module acting as a standby. A matching circuit continuously compares the outputs of two modules and interprets any mismatch as a fault in either of the modules or in the matching circuit itself. After the detection of a mismatch, diagnostic programmes are run to locate the fault. If the fault is in a module, it is taken off-line and the normal operation is resumed as a simplex system. The reliability of a duplex system is increased if a faulty module is repaired and returned to operation (repairable duplex system). Examples of duplex configuration are the Bell Electronic Switching Systems (ESS) [To 78], and the AXE telephone exchange system [OsJo 80]. In the No.1 Electronic Switching System (ESS) the reliability objective is that the system down time should not exceed two hours over its forty years life. Another example of dynamic redundancy is a multi processor system proposed by IBM , in [IBM 80].

3.6.6 - Hybrid Redundancy Technique

The combination of static redundancy and Dynamic redundancy is called Hybrid redundancy. In this case the core of system is TMR (or

NMR in general) with a voting circuitry and also there are a few spares which can replace any faulty module in the core [GoGrLe67] [BoCaRo67] [MaAv70].

Fig 3.3 shows this configuration.

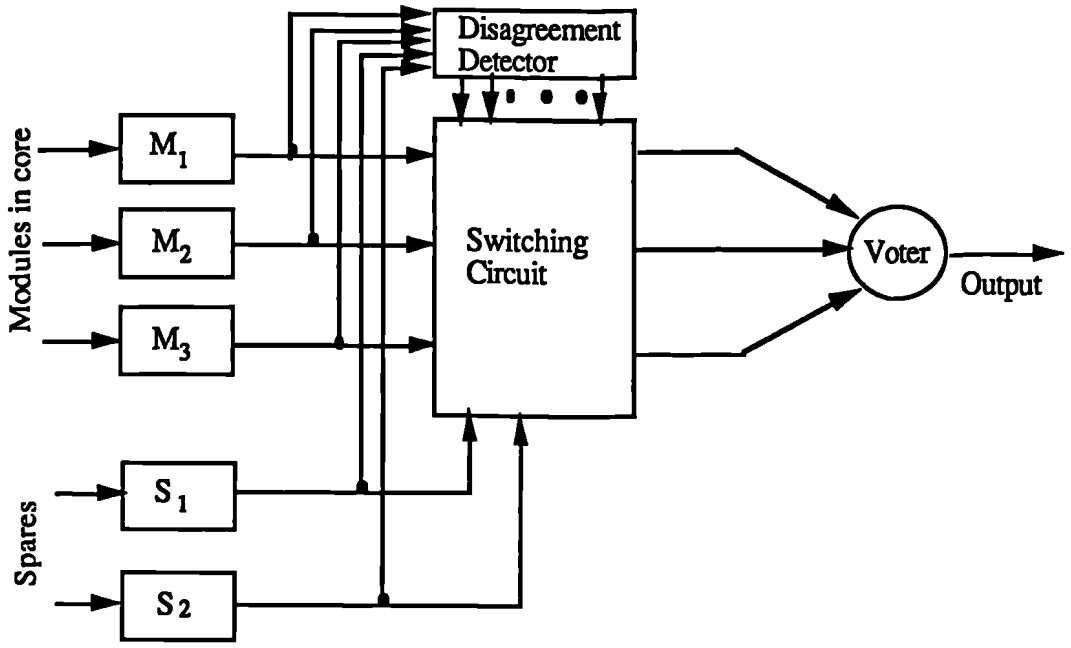


Fig . 3.3 A hybrid structure with a TMR core and two spares

The "disagreement detector" (D) detects whether the system output is different from the output of any TMR module. If there is any failure in TMR, then the faulty module will be replaced by a good spare module by the switch. If all the spares are used up, we will have a TMR system. The maximum number of fault that can be tolerated simultaneously is

n = { (N-1) / 2 } N is the number of modules in the core

e.g. with a TMR core and 2 spares the system can tolerate only one error

$n = (3-1) / 2 = 1$

3.7 - Choosing between TMR and Standby Sparing

Designing systems for standby sparing environments is far more difficult than designing for TMR, since the extra checking and reconfiguration circuitry of sparing is far more complex than the voting circuitry used in TMR.

TMR is often the best in short missions

Hybrid is better for longer missions

3.8 - Responsive Redundancy Techniques

There is a fourth group of hardware redundancy structures which have been proposed in the last 2 decades and do not quite fit in any of the three categories already mentioned [SoMa78]. In the static redundancy structure there is a voting mechanism that votes on the outputs of its N channels, and there is generally no detection and switching circuits. In standby sparing and hybrid techniques there should be a few spare units ready to replace any active faulty module. The spares may be passive before replacement. In a responsive redundant structure all modules are active at the beginning of the mission time. However, there is a detection and switching mechanism to detect and reconfigure

the structure upon the occurrence of a failure so that the contribution of the faulty module is reduced or eliminated. In the following sections we describe a few of this class of redundancy schemes.

3.9 - Self-Purging Redundancy Scheme

A Self-purging redundancy structure is formed from a set of P identical modules, a disagreement detector and a very simple switch for each module[Lo76]. The outputs of these switches are connected to a threshold voter with a threshold M and a weight of one for each input. Fig 3.4 shows a block diagram of a basic self-purging system.

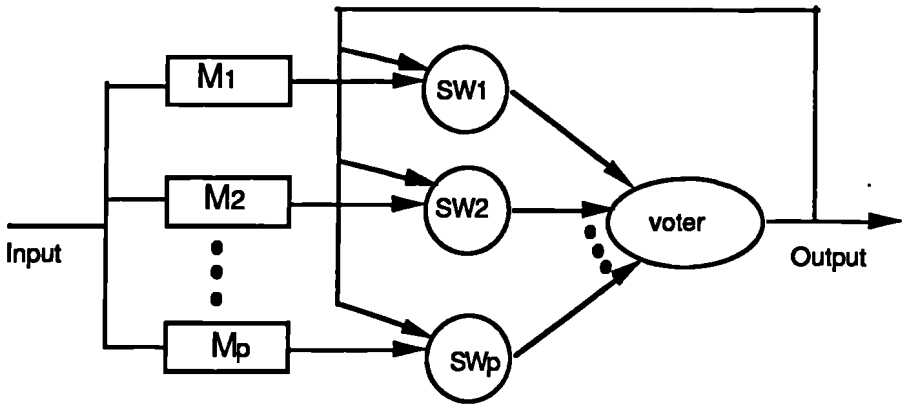


Fig. 3.4 Self-Purging Redundancy Scheme

The voter is a threshold voter and it produces logic 1 if the weighted sum of its inputs is equal to or greater than the threshold level M. It produces logic 0 otherwise. When a failure occurs at the output of a module, this can be detected by the disagreement detectors

implemented in the switch elements. Then a logic value of '0' will be assigned to the output of the faulty module. This is logically equivalent to disconnecting the faulty module from the voter. Suppose that we have a self-purging system with five modules and the threshold of the voter is 3 (M = 3). The voter output Z is

$$Z = m_1[m_2(m_3+m_4+m_5)+m_3(m_4+m_5)+m_4m_5]$$

$$+ m_2[m_3(m_4+m_5)+m_4m_5]+m_3m_4m_5 \qquad 3.1$$

Where $m_1 \ldots m_5$ are the outputs of the modules. The system will operate properly if there are three or more fault-free modules.
The switching system for the self-purging structure is very simple: this is an advantage of this scheme over the hybrid technique. It consists of an EXCLUSIVE-OR gate, a flip-flop, and an AND gate for each module. The switch is shown in Fig . 3.5 .

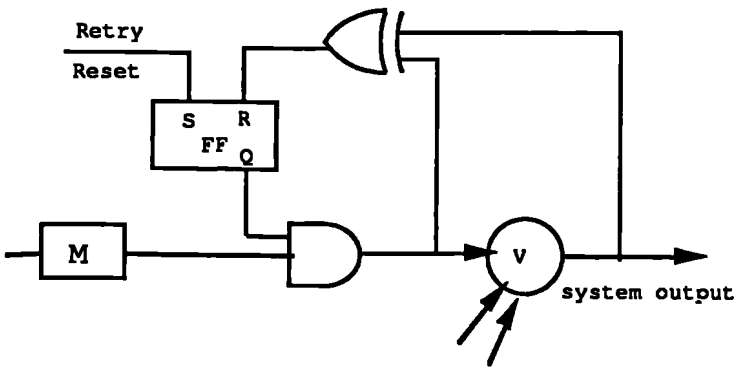


Fig . 3.5 A switch of self-purging scheme

When the first failure occurs, the EX-OR will detect the disagreement between the voter output and the AND gate output, and resets the flip-flop at the next clock pulse and forces the output of the AND gate to be stuck-at-0 . That is the module will not contribute to the voter any more. After the first disagreement between the voter and a module's output, the module will be retried by setting the flip-flop. Therefore if the failure is not permanent, the module can be used again, otherwise it can be removed and repaired or replaced by a good module. This is a very simple switching circuit. System reliability increases as the complexity of switching circuit decreases. The complexity of each individual switch does not increase as the number of modules increases, but the voter complexity increases. This is reversed in systems using hybrid structure. In hybrid systems the voter complexity does not increase (as long as the number of modules in the core is fixed), as the number of modules increases but the complexity of the switching circuit increases rapidly with the number of spares. The disadvantage of self-purging systems over hybrid systems is that , in self-purging all modules are active, therefore, power consumption is higher, also the probability of active module failure is higher than the probability of failure of passive modules (spare modules in hybrid systems). That is

$\lambda > \mu$ where λ is failure rate for active modules and
μ is failure rate for passive modules

However for missions ranging from on-tenth to a few tenths of module mean-life, self - purging redundancy is the best solution. Because for such small missions, the beneficial effect of large dormancy factors can not be taken advantage of significantly. Self - purging systems also can be made to tolerate more multiple failures.

For large mission times (T > 1), stand-by systems are more successful due to the use of dormancy factors for its spares. But of course for very short missions NMR is always the best choice, because there is no switching and detection circuits in NMR.

3.10 - Sift - Out Modular Redundancy

In this structure the system consists of L identical channels, where L can be any integer. The channels are synchronised with one another and perform simultaneous operations[SoMa78]. A comparison is made at the channels' outputs. If a channel fails, its output will be different from the other channels' outputs and it will be sifted out and its contribution to the system output will be terminated. Then the system becomes an (L - 1) redundancy scheme. If another failure occurs, the process repeats itself. This structure can tolerate up to L - 2 failures. That is if L - 2 channels fail, the system still will operate correctly. But when the number of channels reduces to two and another failure occurs, the system can not detect it.

Figure 3.6 shows the block diagram of this structure.

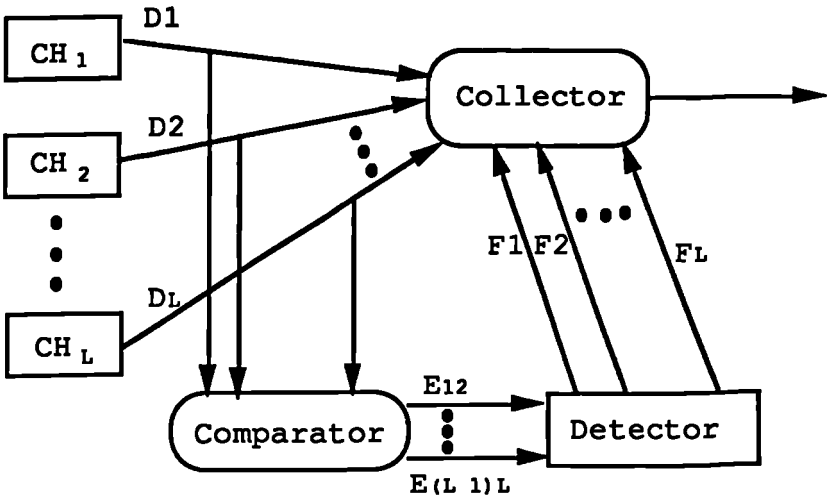


Fig . 3.6 The block diagram for the Sift Out Redundancy Scheme

The comparator is a set of EXCLUSIVE OR gates which check the outputs of the channels against each other. The comparator circuit diagram for three channel SMR is shown in Fig. 3.7 .

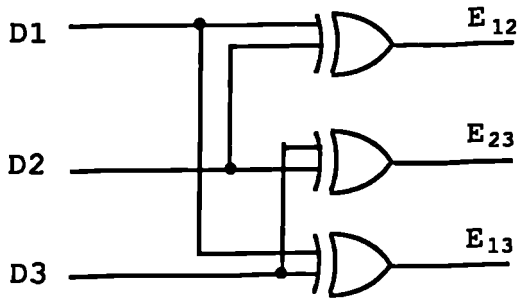


Fig 3.7 The Comparator circuit diagram (L = 3)

The detector is implemented by NOR gates and flip _ flops ,as shown in fig. 3.8 for a three channel SMR. The number of NOR gates is

$[\binom{L}{2} + L]$.

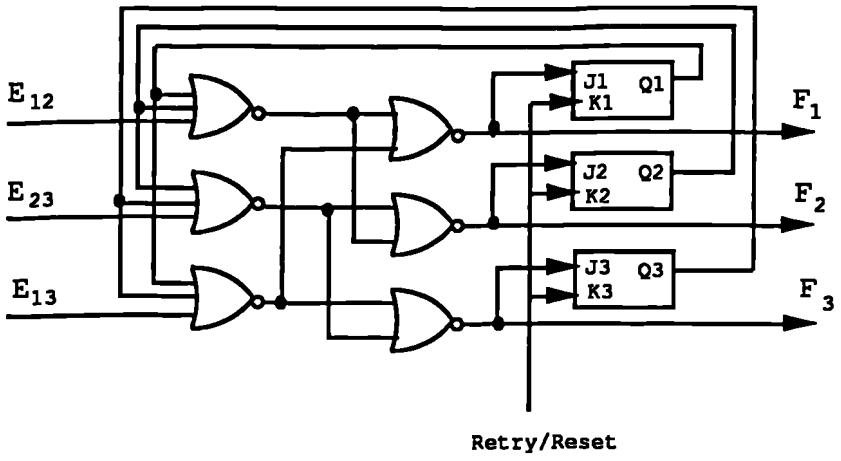


Fig . 3.8 The logic diagram of the Detector circuit (L = 3)

The flip -flops provide the reset / retry facilities. Suppose that channel i is fault free at time t, then F would have the logic value of 0 . But when this channel fails F becomes 1 . For example if channel 1 fails, the EXCLUSIVE - OR's detect the disagreement and cause the lines E12 and E13 to be logic 1 . Then line F1 will be stuck-at-1 and the corresponding flip-flop will force it to hold the logic 1 . The retry procedure makes the structure tolerant to temporary faults. The next block in this structure is the collector. Its logic diagram is shown in Fig.3.9. It consists of (L + 1) NOR gates. Each fault-free channel provides a $\overline{D}_i$, and each faulty channel provides a logic 0 input to the

last NOR gate in this block.

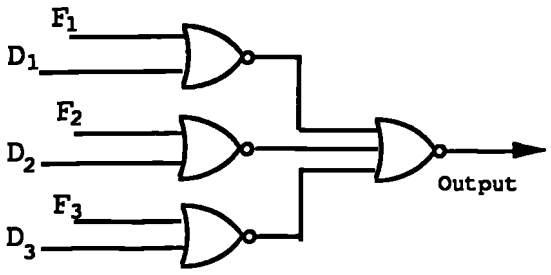


Fig. 3.9 The Collector logic diagram for L = 3

The output of the system is the output of this gate and it is correct if at least two of the L channels are operating correctly.

3.11 - Comparison with other systems

3.11.1 - Comparing SMR with TMR

A Sift-out Modular Redundancy system with three channels has the same fault tolerance as a TMR system. Because at least two channels are required to operate correctly to produce the correct output, there is no need to sift a failed channel out, and therefore the three channels sift out structure can be simplified as shown in fig. 3.10 . This scheme has the capability of automatic fault diagnosis. For example if channel i fails, the output variable F in the detector circuit is set to logic 1 .

This is an important advantage over TMR in commercial computers,

because it shows which módule is faulty and eases maintenance and improves the availability of the system.

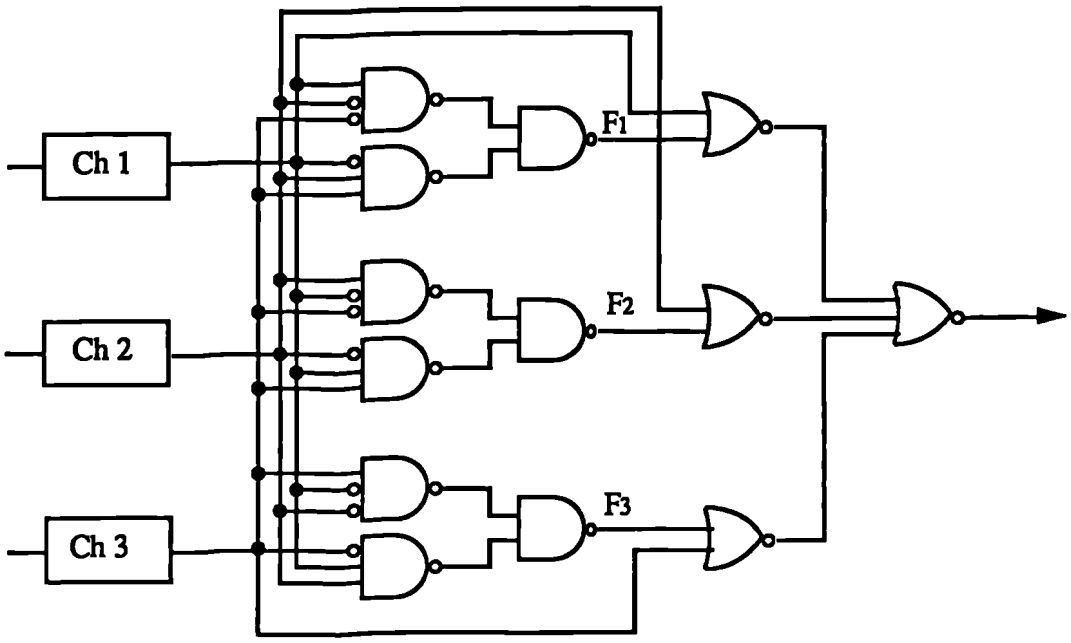


Fig . 3.10 Sift-out redundancy with three channels

3.11.2 - Comparing with NMR

The fault tolerance of an SMR with N channels is (N - 2), but the fault tolerance of an NMR system is (N - 1) / 2 . When N is small, the voter for NMR is less complex than sift - out checking unit, but the situation is the reversed as N increases.

3.11.3 - Comparing with hybrid scheme

Sift-out redundancy has the same fault tolerance or even higher fault tolerance than hybrid configuration. We will compare their reliabilities in the next chapter.

3.11.4 - Comparison with self-purging technique

When L, the number of channels is small the sift-out structure can be implemented with less gates and elements than the self-purging system. For example, a three channel self-purging structure needs three flip-flops, three EXCLUSIVE - OR gates and seven elementary gates, where the SMR requires only thirteen elementary gates. For L > 3 the self - purging has a less complex switch.

3.12 - Summary

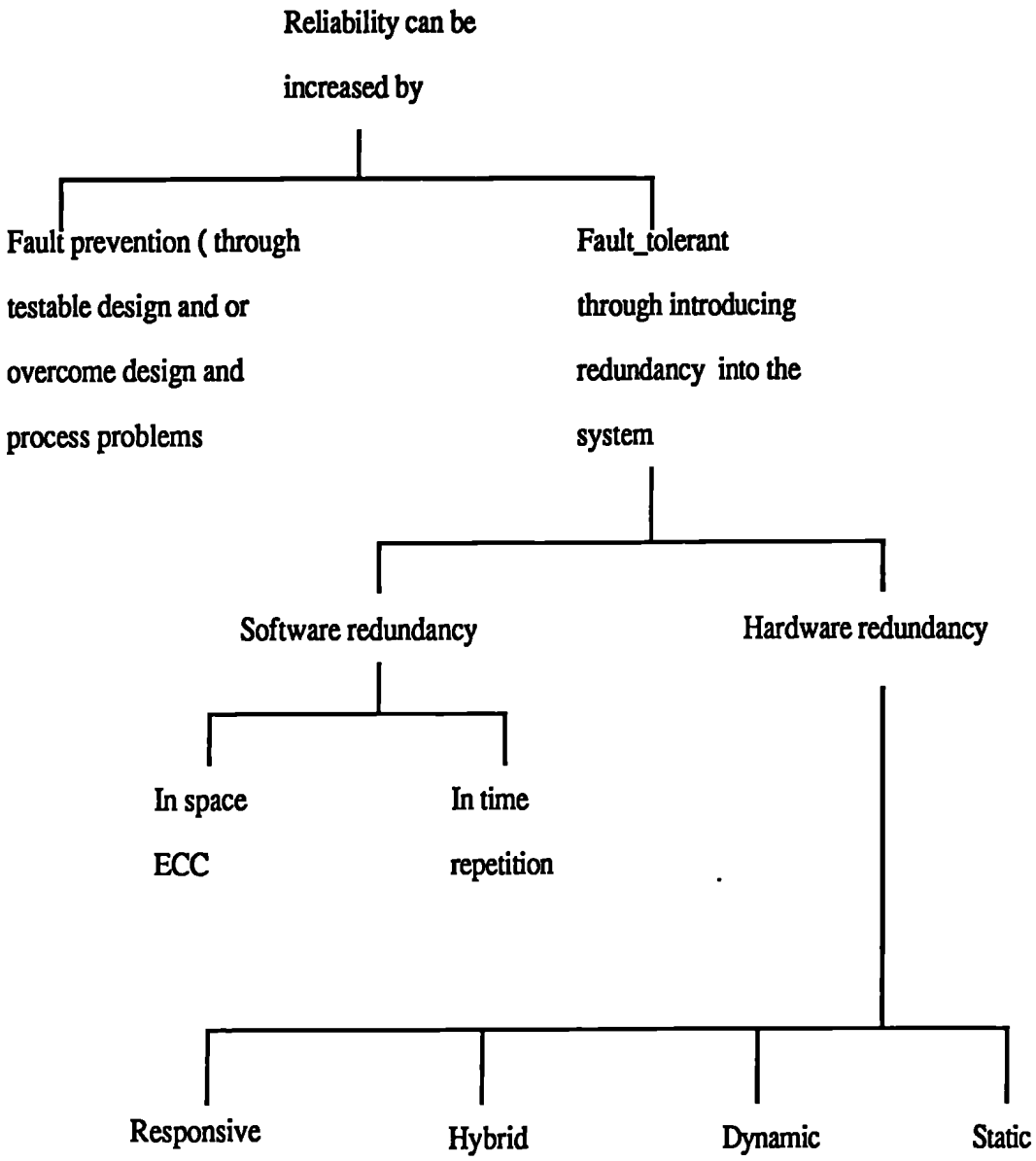
To summarise this chapter, reliability of computing systems may be enhanced by fault prevention (i.e. elimination of faults through design and manufacturing considerations) or by employing redundancy to tolerate both the faults which are built into the system and become active during its operation and those which develop later. Fault-tolerant computing is in no way a new discipline (it began in the design of the earliest computers), and is not an alternative to fault

prevention but is a complement to it.

Ultra-high reliability and high availability (which are in great demand) can be achieved by fault-tolerant techniques. The high costs of testing complex circuits, the relatively low costs of VLSI devices, and the advances in semiconductor technology (which has reduced the size of electronic components and has made room for the application of redundant hardware) have all contributed to making the development of fault-tolerant techniques an alternative way to achieve high reliability.

After a discussion of fault-tolerant principles, a classification of redundancy techniques was presented. This is summarised on the table in the next page.

Finally the most important redundancy techniques (such as TMR, dynamic, hybrid, and responsive redundancy structures) were discussed for the purpose of comparisons with the new designs which will be presented in the next chapter.


Reliability can be increased by
Fault prevention (through testable design and or overcome design and process problems
Fault_tolerant through introducing redundancy into the system
Software redundancy
Hardware redundancy
In space ECC
In time repetition
Responsive
Hybrid
Dynamic
Static

CHAPTER FOUR

NEW FAULT TOLERANT DESIGNS

4.1 - Introduction

In this chapter two new fault-tolerant techniques will be proposed , but first a fault-tolerant structure which has recently been proposed will be discussed and improved upon.

As shown in the earlier chapters Triple Modular Redundancy (TMR) is the best scheme for short mission times (e.g. a few tenths of the MTBF) , but not necessarily for situations in which high reliability is required through out a long life time or at the end of a mission time. For long missions, structures with higher numbers of modules may be more appropriate. Dynamic or hybrid structure often provide better service but they run a cost and area overhead. One of the redundancy techniques which performs remarkably well is a hardware reconfiguration scheme proposed by Su and DuCasse [SuDuCa80]. The structure and behaviour of Su and DuCasse design with 5 modules will be discussed and compared with other five channel redundancy structures . This configuration does not operate correctly with seven modules (contrary to the claim made by the authors). We will introduce an improved version this structure which has fewer gates and operates correctly even with seven redundant modules. An investigation for the maximum number of failures that can be tolerated by each structure will be made.

Fault-tolerant schemes may be divided into two classes when switching and reconfiguration mechanisms are required. In the first group the

complexity of the voter is constant, but the switch complexity is increased significantly (exponentially) with the number of modules in the system. Examples of this class include hybrid structures and basic dynamic structures without voters. In the second group the switch complexity is increased linearly with the number of modules, but the voter complexity is increased exponentially. An example of this class is the Highly Reliable - Highly Efficient structure which will be presented in this chapter. As the structure of the voters in systems using fault-tolerant techniques are more regular and straight forward than the structure of the switches in these systems, and as it is possible to implement highly reliable voters, using space-efficient techniques such as symmetric switching circuits, the second group seems more promising than the first. Two designs in the second group will be proposed in the last sections of this chapter and their advantages and disadvantages will be discussed. The reliability calculation for these structures will be discussed in the later chapters.

4.2 - 5MR Reconfigurable Scheme

In this section we describe a reconfiguration scheme [SuDuCa80] which can tolerate a double fault followed by a single failure; this can be tolerated by neither a 5MR nor a Hybrid (3,2) redundant system with a TMR core. In the next section we will describe some improvements on this design.

Let us consider a system with 5 identical modules (5MR). The outputs of these modules are binary variables x_i 's (i=1,...,5) which are connected to a majority gate. The output of the majority gate is given by equation (4.1).

$$Z = M(X_1, X_2, X_3, X_4, X_5)$$

$$= X_1(X_2(X_3 + X_4 + X_5) + X_3(X_4 + X_5) + X_4 X_5) +$$

$$(X_2(X_3(X_4 + X_5) + X_4 X_5) + X_3 X_4 X_5 \qquad (4.1)$$

substituting $X_1=0$ and $X_2=1$ into the above expression, we obtain

$$Z = M(0, 1, X_3, X_4, X_5) = X_4 X_5 + X_3 X_4 + X_3 X_5$$

$$Z = M(X_3, X_4, X_5) \qquad (4.2)$$

Therefore a TMR system can be obtained by substituting any variable by 0 and any other variable by 1. If we substitute 0 and 1 for the appropriate variables when a single or double fault occurs, the system will be reconfigured to a TMR system. A block diagram for a system that can do these substitutions (and reconfigure the system) is given in Fig. 4.1.

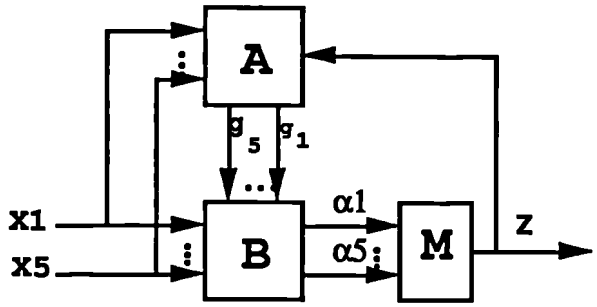


Fig. 4.1 The structure of the 5MR reconfiguration scheme

Block A consists of five "equivalence detectors" and five SR flip-flops. Each detector has two inputs x_i and z and one output g_i, where i=1,2,...,5. Suppose that one of the modules, say module i is faulty, then the logical value of the voter output (z) and the module output x_i will be different, and therefore g_i will have value 1. The logic diagram of block A is shown in Fig. 4.2.

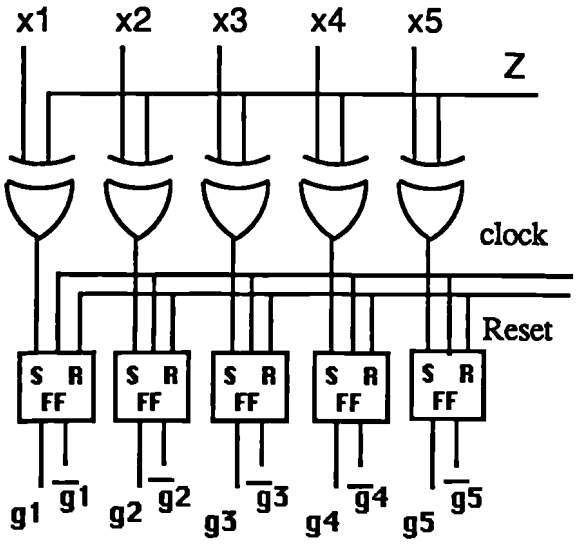


Fig. 4.2 The Disagreement Detection circuit (Block A).

Initially all the RS flip- flops are reset. If all the modules are fault free, $g_i = 0$ (i=1,...,5) the network B transmits X_i's to α_i's for (i=1,...,5).

Block B is shown in Fig. 4.3. It consists of 5 sub-blocks B_i (i=1,5) and each sub-block Fig. 4.4, consists of 5 gates. The design of block B is such that if module i is faulty $g_i = 1$ and α_i becomes stuck-at-0 (s-a-0). Then α_{i+1} (mod. 5) is forced to become s-a-1 (stuck-at-1) temporarily and the remaining α_i's being unaffected by the flip- flops, allow the X_i to transfer to α_i.

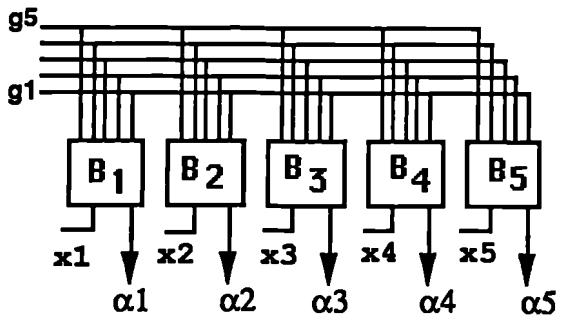


Fig. 4.3 The Switch (Block B) which consists of five sub-blocks B_i.

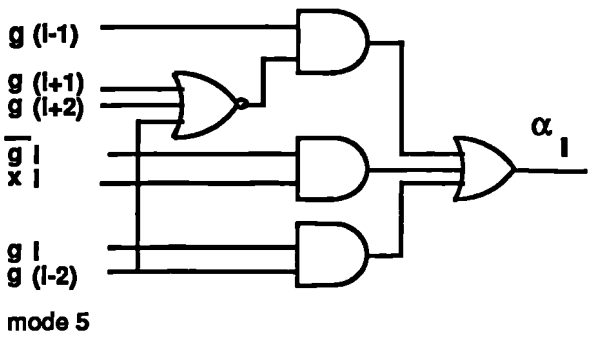


Fig . 4.4 The logic diagram for each switch element (Bi).

The expression for α_i is shown in equation 4.3.

$$\alpha_i = g_{i-1} \sum_{k \neq i-1,\, i} \bar{g}_k + X_i \bar{g}_i + g_i g_{i-2} \qquad (4.3)$$

To verify the above equation for single and multiple faults let us consider the following cases:

Case 1 - System with a single fault

Suppose that module 1 is faulty. Hence $g_1 = 1$ and $g_i = 0$ (i=2,5), substituting these values into equation (4.3) we obtain

$$\alpha_1 = g_5(\overline{g_2 + g_3 + g_4}) + X_1 \bar{g}_1 + g_1 g_4$$

$$\alpha_1 = \quad 0 \quad + \quad 0 \quad + \quad 0$$

$$\alpha_1 = 0, \quad \alpha_2 = 1 \text{ and } \quad \alpha_1 = X_i \qquad \text{for } i = 3, 4, 5$$

Thus the 5MR system is reconfigured to a TMR with module 2 as a spare.

Case 2 - System with Double fault

There are two possibilities to be considered in this case:

a) Non-adjacent double fault.

If modules 1 and 4 are faulty, then $g_1 = g_4 = 1$ and $g_2 = g_3 = g_5 = 0$. Using equation (4.3) we obtain $\alpha_1 = 1$, $\alpha_2 = X_2$, $\alpha_3 = X_3$, $\alpha_4 = 0$, and $\alpha_5 = X_5$. Therefore, the system reconfigures into a TMR.

b) Adjacent double fault.

If modules 1 and 2 are faulty, then $g_1 = g_2 = 1$ and $g_3 = g_4 = g_5 = 0$. Substituting them into equation (4.3), we obtain $\alpha_1 = 0$, $\alpha_2 = 1$, and $\alpha_3 = X_3$, $\alpha_4 = X_4$, $\alpha_5 = X_5$. The system reconfigures to a TMR again.

Case 3 - System with Treble fault.

If the third module now fails, the system will change the majority gate in the TMR to an OR or an AND gate. This is shown below.

a) If module 3 fails after module 1 and 4 have already failed, then $g_1 = g_3 = g_4 = 1$ and $g_2 = g_5 = 0$, using the equation 4.3 we obtain

$\alpha_1 = 1$, $\alpha_2 = X_2$, $\alpha_3 = 1$, $\alpha_4 = 0$, and $\alpha_5 = X_5$. hence $Z = X_2 + X_5$

that is the majority gate is changed to an OR gate and therefore the

fourth failure can be tolerated if it is s-a-0

b) Similarly if module 2 or 5 fails the TMR system will be reconfigured to an AND gate. Then the fourth failure can be tolerated if it is s-a-1.

4.3 - Comparison with other similar schemes

The above scheme operates well when the number of modules is five . Table 4.1 shows a comparison between the number of faults that can be tolerated by this scheme , a five modular redundancy (5MR) , and a hybrid with a TMR core and two spares H(3,2) . The circuit realisation of this structure is simpler than many other similar designs .

Faults	Effect on proposed Scheme	Effect on 5 M R	Effect on Hybrid (3 , 2)
1	Reconfigure to TMR	None	None
2 simultaneously	Reconfigure to TMR	None	Failure
2 in sequence	Reconfigure to TMR	None	None
2 simultaneously, followed by 1	OR gate or AND gate, OK	Failure	Failure
3 in sequence	OR gate or AND gate, OK	Failure	None
1 followed by 2	Failure	Failure	Failure
3 simultaneously	Failure	Failure	Failure
4 after surviving 3	OK if AND gate	Failure	failure
s - a - 1	(50% of time)		
s - a - 0	OK if OR gate (50% of time)	Failure	Failure

Table 4.1

The number of gates used by the detection circuit and the switch is less than either the iterative cell switch scheme presented by [SiMc73], or self-purging redundancy [L076]. The structure of the switching

mechanism is highly regular, as all the sub-blocks Bi's shown in Fig. 4.3 are repeated for i from 1 to 5 as are the disagreement detection circuits. This regularity is another advantage of the scheme , as it reduces the manufacturing cost , and makes the testing of the circuit easier . However the complexity of the switching circuit , and the voter, increases with the number of modules (N > = 7) .

In the next section an improved version of Su and DuCasse's scheme will be presented .

4.4 - Design Improvement

The design improvement is proposed for the following reasons.
I - Anticipating the implementation of the logic with hardware however, it is important to implement the circuit requiring fewest number of gates. As a rule in digital circuits, if we have fewer gates, the cost will be lower, the yield will be higher, the heat produced will be less, the area used will be less, the power consumption will be reduced, and the reliability will be increased. Therefore we propose another scheme which uses the same idea as the 5MR reconfigurable system but requires fewer gates for the implementation of block B. In the case of a single fault there is no need to force the output of one of the fault free modules to be s-a-1. Only in the case of a double fault, the output of one of the faulty modules must be s-a-0 and the output of the other faulty module must be s-a-1. Then the reconfigured system is a perfect

TMR and can tolerate the next failure, and then the majority gate becomes an AND gate or an OR gate after occurrence of another failure.

II - With reference to the equation 4.3 , the Su and DuCasse's scheme develops a problem when the number of modules is seven (N = 7), and multiple failures occurring as in the following example:

Let us assume that there are seven identical modules , and modules m1 , m2, and m5 fail , Thus

$g_1 = g_2 = g_5 = 1$ and $g_3 = g_4 = g_6 = g_7 = 0$.

Substituting these values into the equation 4.3 , we obtain

$$\alpha_1 = g_7(\overline{g_2 + g_3 + g_4 + g_5 + g_6}) + X_1\bar{g}_1 + g_1 g_6$$

$\alpha_1 = 0 + 0 + 0 = 0$ also $\alpha_2 = \alpha_5 = 0$

$\alpha_3 = X_3$, $\alpha_4 = X_4$, $\alpha_6 = X_6$,and $\alpha_7 = X_7$

Now if the fourth failure happens to be stuck-at-0 , the output of the majority gate will be 0 , irrespective of the output values of the other fault free modules, since there are already three inputs of 0, to the voter.

Therefore the 7MR Reconfiguration structure can not tolerate more than three failures if they occur as above. It should be noticed that a basic 7MR without any extra mechanisms for switching and

disagreement detection circuits will tolerate any three failures. Hence the validity of the scheme presented by Su and DuCasse is called into question when the number of modules is greater than five.

However in the improved scheme , α_i is computed by equation 4.4 . The switch will be implemented with fewer gates than we used in the original design suggested by Su and DuCasse,

$$\alpha_i = g_i g_{i-1} \overline{g}_{i+2} + x_i \overline{g}_i + g_i \, g_{i-2} \overline{g}_{i+1} \qquad (4.4)$$

The logical diagram of sub-block B_i is shown in Fig. 4.5.

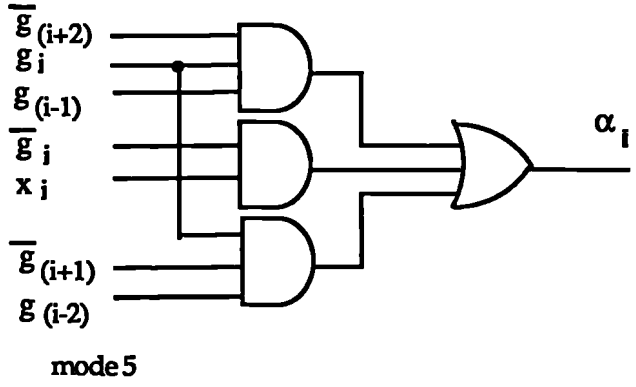


Fig. 4.5 The switch element for the improved version.

The NOR gate and one of the AND gates in Fig. 4.4 are replaced by only a single three input AND gate in Fig. 4.5.

Now to verify equation (4.4) , we consider the following cases :

- System with a single fault

Suppose that module 1 is faulty, then $g_1=1$ and $g_i=0$ for$(i=2,5)$.

Substituting these values in equation (4.4) we obtain:

$\alpha_1 = g_1 g_5 \overline{g}_3 + X_1 \overline{g}_1 + g_1 g_4 \overline{g}_2$ $\qquad \alpha_1 = 0 + 0 + 0$

$\alpha_1 = 0$, and with the same procedure $\alpha_2 = X_2$, $\alpha_3 = X_3$, $\alpha_4 = X_4$,

and $\alpha_5 = X_5$.

Thus the system reconfigures to a 4MR.

$Z = M(0, X_2, X_3, X_4, X_5)$

- System with double fault.

a) Non-adjacent double fault.

Suppose modules 1 and 4 fail. Then substituting $g_1 = g_4 = 1$ and

$g_2=g_3=g_5=0$ into equation (4.4) we obtain

$\alpha_1 = g_1 g_5 \overline{g}_3 + X_1 \overline{g}_1 + g_1 g_4 \overline{g}_2$ thus $\alpha_1 = 0 + 0 + 1 = 1$

and $\alpha_4 = g_4 g_3 \overline{g}_5 + X_4 \overline{g}_4 + g_4 g_2 \overline{g}_5$ so $\alpha_4 = 0 + 0 + 0 = 0$

Similarly $\alpha_2 = X_2$, $\alpha_3 = X_3$, and $\alpha_5 = X_5$.

Therefore the system successfully reconfigures to a TMR.

b) Adjacent double fault

If modules 1 and 2 fail, i.e. $g_1 = g_2 = 1$ and $g_3 = g_4 = g_5 = 0$, with the same procedure $\alpha_1 = 0$, $\alpha_2 = 1$, $\alpha_3 = X_3$, $\alpha_4 = X_4$, and $\alpha_5 = X_5$. So $Z = M(0, 1, X_3, X_4, X_5)$, (see equation 4.2). The system changes to a perfect TMR.

- **System with treble fault**

Suppose the third module fails. For example if module three fails after failure of module 1 and 4 in case (2-a), then $g_1 = g_3 = g_4 = 1$ and $g_2 = g_5 = 0$.

By using equation (4.4) we obtain:

$$\alpha_1 = 1,\ \alpha_2 = X_2,\ \alpha_3 = 0,\ \alpha_4 = 0, \text{and}\ \alpha_5 = X_5$$

Also from equation (4.1), we obtain $Z = X_2 . X_5$, that is the majority gate is changed to an AND gate and the fourth failure can be tolerated if it is stuck-at-1 . Similarly if module 2 or 5 fails we get $Z = X_3 . X_5$ or $Z = X_2 + X_3$ respectively , which similarly changes the majority gate to an AND or a OR gate respectively and hence , the fourth failure can be tolerated if it is stuck-at-1 or stuck-at-0 respectively .

All the possible cases have been investigated for both N = 5 and N = 7 modules and it seems the above realisation will function correctly with all the failure cases discussed in this chapter . Appendix A will show the results.

In the schemes discussed so far in this chapter ,with N = 5 , the number of faults that can be tolerated is greater than 5MR (five Modular Redundancy), and Hybrid(3,2) (a TMR core and two spares). If only one of the modules fails, the system reconfigures to a TMR in the scheme proposed by Su and DuCasse and to a 4MR in the improved version . Then they can tolerate the 3rd failure and the majority gate is changed to an AND gate or an OR gate. The fourth failure can be tolerated half of the time.

In the improved version though the structure of the switch is more regular , it uses fewer gates than the switch in the scheme proposed by Su and DuCasse [SuDuCa80]. Only four gates (three AND gates and one OR gate) are used for each sub-block B_i , compared to five gates used in Su and DuCasse's scheme.

Another important result is that the improved version of the structure discussed above offers higher fault-tolerance than the original scheme by Su and DuCasse with N = 7 .

4.5 - Multiple Fault-Tolerant Reconfigurable Structure

This section presents a new fault-tolerant structure [HaZi89] for increasing system reliability and fault-tolerance by adding redundant hardware. In this scheme a new switching circuit is designed. The system operates with very accurate timing . A D-type flip-flop is used to synchronise the circuit and prevent oscillation. An equation for the switch function will be derived and verified in the case of five modules and single or multiple failures.

4.5.1 - The Realisation of the Proposed Structure

The proposed scheme can be realised with N identical modules, a switching circuit , a disagreement detection circuit, and a majority voter . The structure reconfigures itself after the occurrence of failures and it can tolerate multiple faults which will be described later. The block diagram for the Multiple Fault-Tolerant Reconfigurable Structure (MFT-RS) with five modules is the similar to the 5MR reconfigurable scheme shown in Fig. 4.1 .

Let us consider such a system as mentioned above. The outputs of the five modules are binary variables (x_1 to x_5)., as in the previous structures. They are connected to a majority gate, the output of the majority gate will be given by equation 4.1 , as defined in section 4. 2 The disagreement detection circuit consists of five "equivalence

detectors" and one D-type flip-flop. Each detector has two inputs x_i and Q which holds the logical value of the voter output (z), and one output d_i , where i=1,2,...,5. Thus the structure of the disagreement detection circuit is similar to the detection circuit (Block A) in the 5MR reconfigurable scheme discussed in section 4.2 The difference is the application of the flip flops . Instead of the five R-S flip flops in the 5MR reconfiguration scheme, only one D-type flip flop is used as shown in Fig. 4.6. The advantage is that N, R-S flip flops are replaced by only one D-type flip flop. However logic value 1 will appear at the output of any EXCLUSIVE- OR (d_i) when a disagreement occurs between the system output (Z) and the module output (X_i) . The initial requirement is that the flip flop and X_i's are set to 0. When all the modules operating correctly , $d_i = 0$, for all values of i from 1 to N. Thus the modules' outputs will be transmitted to the voter without any change, that is $V_i = X_i$ (for i = 1 to i = N). In our description N = 5.

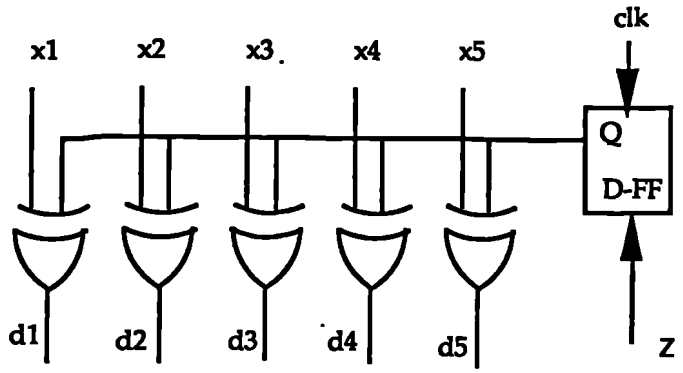


Fig 4.6 The logic diagram of disagreement detection circuit.

Similar to the previous structure , the switch forces the output of the faulty module, which is connected to the voter , to logic 0 . This is logically equivalent to disconnecting a failed module from the voter. That is if module i is faulty $d_i = 1$, $V_i = 0$, and $V_j = X_j$ (for j =1 to N and j is not equal to i) .

The logic diagram of the switch is illustrated in Fig. 4.7 . With reference to Fig 4.7, equation 4.5 may be obtained for the outputs of the switch.

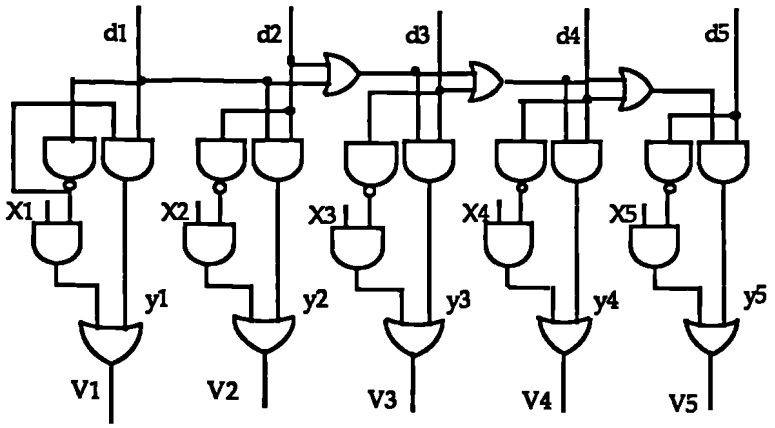


Fig 4.7 The logic diagram for the switch.

$$v_i = \bar{d}_i x_i + y_i \qquad \text{where } y_i = d_i \sum_{j=1}^{i-1} d_j \qquad \text{for } i > 1 \text{ and } y_1 = 0 \qquad (4.5)$$

Let us consider our familiar cases as before to verify the above equation. That is testing the structure with a single , or multiple faults.

Case 1 System with a single fault

Suppose that module i is faulty. Then $d_i = 1$ for (i=1,...,5) and $d_m = 0$ for (m=1,...,5 and not equal to i), substituting these values into equation (4.5) we obtain

$v_i = 0$, $v_m = x_m$. Thus the 5MR system is reconfigured to a 4MR .

Case 2 System with Double fault

If modules i and j are faulty , then $d_i = d_j = 1$ (for j = 1,...,5 and not equal to i) and $d_m = 0$ for (m = 1,...,5 and not equal to i or j). There are two possibilities to be considered in this case:

using equation (4.5) we obtain

I) $v_i = 0$, $v_j = 1$ and $v_m = x_m$ if i<j or

II) $v_j = 0$, $v_i = 1$ and $v_m = x_m$ if j<i .

Thus the system reconfigures to a TMR.

Case 3 System with Treble fault.

Now if the third module say module k fails after failure of modules i and j, then $d_i = d_j = d_k = 1$ (k = 1,...,5 and not equal to i or j) and $d_m = 0$ for (m = 1,...,5 and not equal to i or j or k) then for k<i and j, $v_k = 0$, $v_i = v_j = 1$ and $v_m = x_m$ otherwise $v_k = 1$ and v_i and v_j

stuck-at-0 and or 1 as in the case 2-I or 2-II . Then the system reconfigures to an OR gate ,therefore it tolerates the fourth failure if it is stuck-at-0.

The conclusion of this scheme is similar to the conclusion of the previous structure described in section 4. 4. After occurrence of the 3rd failure, the majority gate will be reconfigured to an OR gate, therefore a fourth failure will be tolerated if it is a stuck-at-0 fault. It should be noted that the system with five modules (N = 5) can not tolerate three simultaneous failures (this is in contrast to a five modular redundancy technique with a majority voter). Thus the same table as in section 4.3 may be used for comparison with other similar schemes, apart from the first and the last lines which should be

a) With a single fault the system reconfigures into a 4MR

b) The majority gate reconfigures into an OR gate only, in the case of occurring three simultaneous failure. Thus the fourth failure will be tolerated if it is stuck-at-0 fault.

The number of gates used by the above structure is less than any other similar structure discussed so far, because only one flip flop is used in this scheme. The complexity of the switching circuit and detection circuit increase linearly with the number of modules (N>= 7) but the complexity of the voter increases significantly .

4.6 - A Highly Reliable Highly Efficient Hardware Redundancy Structure

In this section another fault-tolerant design will be presented [Ha89] which shows some advantages over other techniques suggested in the literature, particularly when the number of redundant modules is seven. Before any discussion about this technique let us consider the cases that cause system failure in the other schemes with N = 7.

- Basic 7MR can not tolerate more than three failures .
- 7MR Reconfiguration scheme by Su and DuCasse fails with three modules' failures .
- Hybrid (TMR core, 4 spares) fails in cases of 2 simultaneous failure, or 3 simultaneous failures, or 1 failure followed by 2 simultaneous failures.
- Hybrid (5MR core, 2 spares) not only fails in the event of 3 simultaneous failure , but it has a very complex switch as well.

Introduction of the new redundancy technique which will be presented next, prevents the failure cases which cause system failure in the above mentioned schemes .

The structure consists of N identical modules, a disagreement detection circuit, and a newly developed switch which is connected to a majority voter. All modules are active. In the event of module failures, the switch forces the logical outputs of the odd faulty modules in the

structure to 1 , and the even faulty modules to 0. This process will be continued until two fault free modules remain, therefore this configuration tolerates up to (N - 2) module failures. It may tolerate the (N - 1)th failure if it is stuck at 0 fault. The implementation of the structure is very easy and highly regular . It shows several advantages over the existing redundancy techniques.

4.7 - Realisation of the Highly Reliable Highly Efficient Structure (HR-HE)

To describe the scheme let us consider a system with N identical modules where N is an odd number. The block diagram of this scheme is shown in Fig. 4.8.

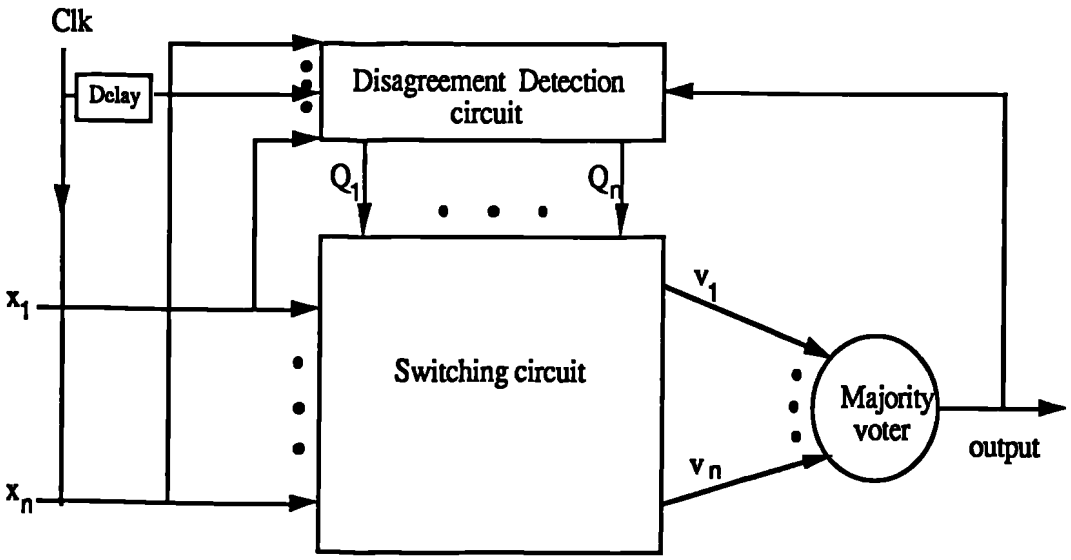


Fig. 4.8 The block diagram for the proposed scheme

The disagreement detection circuit consists of N "equivalence detectors" and is similar to the detection circuit shown in Fig 4.2 for the 5MR reconfigurable scheme.

In this presentation we refer to the module numbers starting from left, as in Fig. 4.9. Initially all R-S flip flops in the disagreement detection circuit are reset. If all the modules are fault-free, $d_i = 0$ (i =1 to n). Therefore

$$Q_i = 0, \quad \overline{Q_i} = 1 \quad (i = 1 \text{ to } n)$$

and the switch transmits x_i's to v_i's and the majority gate outputs the majority of the n inputs.

The important feature of this structure is that the switch forces the odd faulty modules to stuck-at-1 and the even faulty modules to be stuck-at-0. Therefore the output of the majority voter will not be affected when there are one or more pairs of faulty modules in the system. Assume that the voter and reconfiguration circuit are fault-free.

The switch is shown in Fig. 4.9. It consists of N identical switch components as marked by the doted lines in the figure .

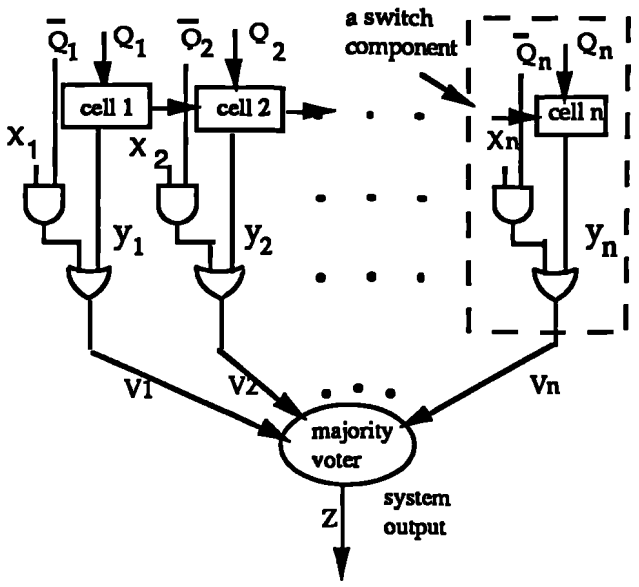


Fig. 4.9 The switch structure

There is one cell in each switch component, and each cell has two inputs (Q_i and K_{i-1} for the ith cell). Fig. 4.10 illustrates the logic diagram for each cell .

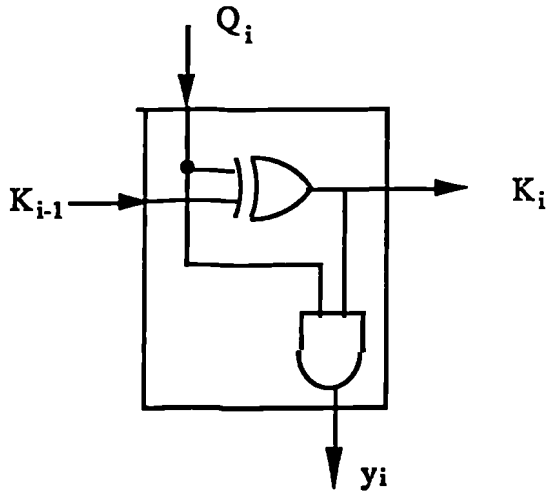


Fig. 4.10 The cell logic diagram

The Q input indicates whether the related module is faulty ($Q_i = 1$), or fault free ($Q_i = 0$). The other input indicates whether any odd number of failures occurred in the preceding modules ($K_{i-1} = 1$) or not

(K_{i-1}= 0). Therefore there are four possibilities to be considered for each cell. Table 4.2 shows the possible logic values of the outputs for each cell.

Input		Output	
Q_i	K_{i-1}	y_i	K_i
0	0	0	0
1	0	1	1
0	1	0	1
1	1	0	0

Table 4.2 . The table illustrate the behaviour of cell i

Referring to the logic diagram in figures 4.9 and 4.10 K_i and y_i may be obtained by equation (4.6)

$$K_i = Q_i \oplus K_{i-1} \qquad \text{where} \qquad K_0 = 0 \qquad (4.6)$$

$$y_i = Q_i . K_i$$

Finally the outputs of the switch are connected to a majority gate.Using the above equation Vi may be expressed as in equation 4.7 .

$$V_i = Y_i + (X_i . \overline{Q_i}) \qquad (4.7)$$

To verify the above equation for single and multiple faults let us

consider a system with five identical modules using the above structure . However the system should be able to tolerate at least three failures. Consider the following cases:

Case 1 - System with a single fault

Suppose that module 1 is faulty. Hence $d_1 = 1$ and $d_i = 0$ (i=2 to 5), substituting these values into equations (4.6) and (4.7) we obtain

$$K1 = Q1 \oplus K0 = 1 + 0 = 1 \qquad \text{where} \qquad K0 = 0$$

$$y1 = Q1 \,.\, K1 = 1 \,.\, 1 = 1$$

$$\text{and} \quad V1 = Y1 + (X1 \,.\, \overline{Q1}) = 1 + (X1 \,.\, 0) \quad \text{thus} \quad V1 = 1$$

Following the same procedure we obtain

$$V2 = X2, \quad V3 = X3, \quad V4 = X4, \text{ and } \quad V5 = X5$$

Therefore the correct output can be obtained by the majority voter.

Case 2 - System with Double fault

If modules 1 and 3 are faulty, then $d_1 = d_3 = 1$ and $d_2 = d_4 = d_5 = 0$. Using equations (4.6) and (4.7)we obtain $V_1 = 1$, $V_2 = X_2$, $V_3 = 0$, $V_4 = X_4$, and $V_5 = X_5$ .Therefore, the system reconfigures into a TMR

and the correct output is obtained.

Case 3 - System with Treble fault.

Suppose that module 4 fails after failure of modules 1 and 3. Using the same procedure we have $d_1=d_3=d_4=1$ and $d_2=d_5=0$, and we obtain:

$V_1 = 1$, $V_2 = X_2$, $V_3 = 0$, $V_4 = 1$, and $V_5 = X_5$. hence $Z = X_2 + X_5$.

That is the system is reconfigured into an OR gate thus any fourth failure may be tolerated if it is stuck at 0 fault (s-a-0). The reset and retry facilities for the flip-flops may be used to deal with transition failures.

4.8 - Highly Reliable Highly Efficient Structure (HR-HE Structure) compared with other fault-tolerant designs.

HR-HE Structure has a few advantages over the other designs. Supposing that we have a system consisting of five modules as described in the earlier example. If we compare this system with a five Modular Redundant structure (5MR), the number of failures that the HR - HE scheme can tolerate is at least (N-2) i.e. 3, while the number of failures tolerated by 5MR is 2. This is a good advantage particularly when N>5.

Comparison of this scheme with a hybrid consisting of a TMR core and

two spares,H(3,2) (with an iterative switch design) [SiMc73] shows that H(3,2) will fail in the presence of two simultaneous failure but HR-HE structure will survive.

Another advantage of HR - HE structure over many other similar fault tolerant designs is the simplicity of its switch. The number of gates used to implement the switch is less than those used in switches in some other schemes such as the iterative cell array switch in hybrid redundancy, the switch in the 5MR reconfigurable scheme by [SuDuCa80] , and the switch in the sift-out redundancy [SoMa 78].

In the above scheme if only one of the modules fails, the output of that module is stuck at 1, but the majority gate votes correctly on the inputs that it receives. In the case of two module failing the output of the first faulty module is stuck-at-1 and the other stuck at 0. For more module failures this process repeats, until the system reconfigures into a TMR structure . When the system is in the TMR state , it can tolerate at least one more failure , and it reconfigures into an OR gate. Then the next failure may be tolerated if it is s-a-0 fault.

The number of faults that the proposed scheme with N modules can tolerate is greater than that tolerated by NMR (N Modular Redundancy) or Hybrid (3, N-3) [that is a hybrid with a TMR core and N-3 spares]. or the other schemes discussed previously. Diagrams 4.11 and 4.12 show a comparison between this scheme, (using five redundant modules or seven redundant modules), with the other techniques named in the diagrams .

Finally the structure of the switching mechanism is highly regular. Therefore manufacturing cost is less and circuit testing is easier. The complexity of the voter increases significantly when the number of modules is grater than 5 (N > 5), but this is not a disadvantage , as it is possible to implement these majority gates, using symmetric switch arrays on IC`s. Therefore the HR-HE scheme appears to be more effective than similar designs especially when N>5 .

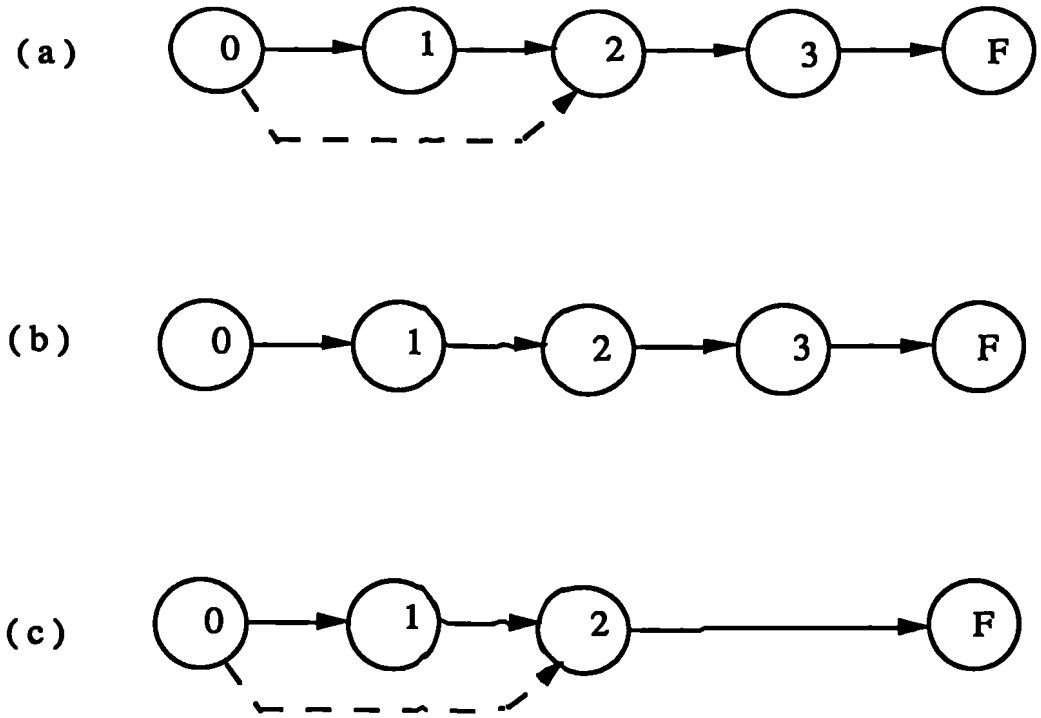


Diagram 4.13 Failure states and transions for 5-tuple redundancy structure,
(a) HR-HE, MFT-RS, and 5MR-Improved Version
(b) Hybrid (TMR core and 2 spares)
(c) Basic 5MR

N.B. In the diagram F denotes complete system failure, numbers represent the number of failed modules, solid arrows represent occurance of single failure, and any case of two simultaneous failures which doen not cause system failure is shown with a dashed arrow. The probability of 3 simultaneous failures occuring is approximately zero, therefore they are not shown in the graphs.

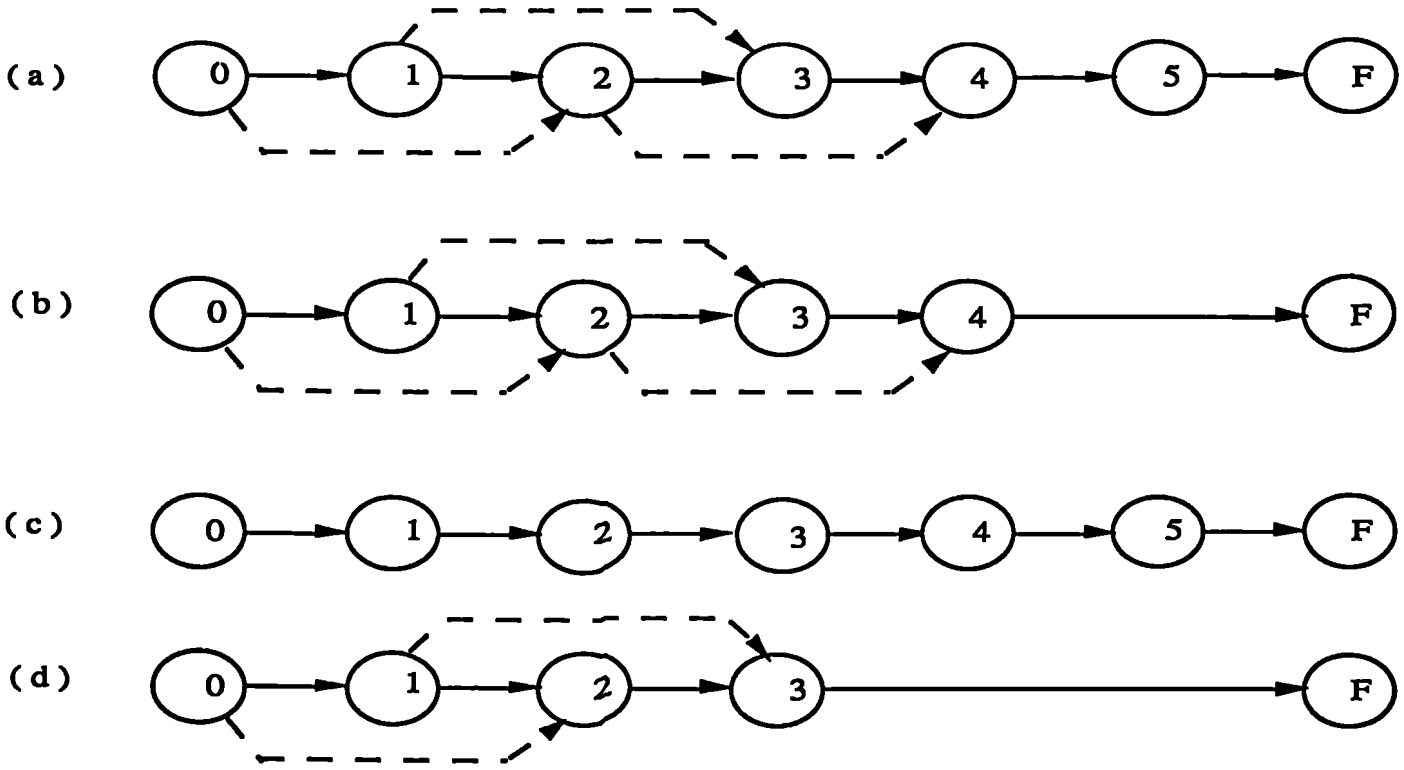


Diagram 4.14 Failure states and transions for 7-tuple redundancy structures,
(a) HR-HE scheme
(b) 7MR-Improved Version , and Hybrid(5MR core and 2 spares
(c) Hybrid (TMR core and 4 spares)
(d) Basic 7MR

N.B. In the diagram F denotes complete system failure, numbers represent the number of failed modules, solid arrows represent occurance of single failure, and any case of two simultaneous failures which doen not cause system failure is shwn with a dashed arrow. The probability of 3 simultaneous failures occuring is approximately zero, therefore they are not shown in the graphs

4.9 - Summary

This chapter has discussed a recently published fault-tolerant technique (5MR Reconfigurable Scheme) [SuDuCa80] and the disadvantages of this technique were shown, particularly when the number of modules is greater than five.

A new design which uses fewer gates in the switch structure than the above scheme was presented. Decreasing the complexity of the switching circuit in a system is a useful way of improving reliability. It was shown that this modified technique has a higher fault-tolerance than the one proposed by [SuDuCa80].

In this chapter two more new techniques were proposed and their advantages over other fault-tolerant designs were discussed. The number of faults that can be tolerated by the different techniques was shown in the two graphs at the end of the chapter and the priority of the new techniques to the others was shown.

CHAPTER FIVE

RELIABILITY COMPARISON OF DIFFERNT TECHNIQUES

5.1 - Introduction

In the previous chapters we have mostly considered techniques to increase the fault-tolerance of systems by the application of various switch designs. Table 5.1 shows the maximum number of faults that can be tolerated by some of the designs discussed earlier.

	TMR	NMR	Hybrid H (N-S, S)	Self-purging	Sift-out scheme	NMR Reconfig.	MFT - RS	HR - HE scheme
Fault-tolerance of systems	1	$\frac{N-1}{2}$	$\frac{(N-S)-1}{2}+S$	$N-M^{*}$	N-2	$\frac{N+1}{2}$	$\frac{N+1}{2}$	N-2
Fault-tolerance of systems for N=7	Not app.	3	5 , S=4 4 , S=3	5 , M=2 4 , M=2	5	4	4	5

* Thereshold of the voter is M

Table 5.1 The maximum number of failures that can be tolerated by the schemes

At this point we should determine whether it is indeed advantageous to increase the fault-tolerance of systems. The parameter that we will consider is the overall reliability of the system. The overall system reliability is an important factor and it is possible to increase the fault-tolerance of a system and yet degrade its reliability due to the addition of a great amount of extra hardware.

However an accurate reliability analysis needs to be done if we are to

compare different systems. For any analysis, including reliability analysis, a mathematical model of the system under investigation is needed. Accurate results, and better understanding of the physical system, can only be gained by good system modelling . There are many reasons for system modelling , including prediction of system behaviour [MaAv70] [Ma71], evaluation of its functions [MaSo75] [Si75], and better control .

5.2 - Reliability modelling and the Assumptions

In the following , a reliability model known as the classical model [BoCaJe71] will be used for evaluation of the reliability of a non-redundant system (with a given failure rate), a TMR , an NMR , and a dynamic structure . The reliability curve for each case will be plotted. Then a new reliability model will be presented and will be applied to the other fault-tolerant designs discussed in the previous chapter.

It is important to know the assumptions made in the model since credibility of the results depends on the model's approximation to the physical system under investigation. In the work described in this chapter the failure phenomena were assumed to be known. The failure rates were also assumed known both with power on and power off. The modelling was limited to the electronic portion of systems. When

required, the behaviour of the system in the presence of faults was assumed known, i.e. the probability of detection and location of the resulting signal errors was assumed known, as well as the probability of successful reconfiguration and recovery.

In addition the following assumptions are made for the classical model:

- exponential distribution is used for the reliability calculation i.e. the failure rate λ, is constant (after burn-in period), thus the reliability of each module at time t, is given by:

$R(t) = \exp(-\lambda t)$

- individual module failures in a system are assumed to be independent, so that the system reliability is the product of the module reliabilities as shown in equation 2.15 (in the case of non-redundant system), or will be evaluated by equation 2.16 in chapter 2 (in the case of a redundant configuration) . For example , in a system composed of n identical

modules or components with failure rate λ_c , the system reliability is

$R_{sys}(t) = (e^{-\lambda_c t})^n = e^{-\lambda_{sys} t}$, where $\lambda_{sys} = n \lambda_c$.

- once a module has failed, its output is assumed to always be in error
- a voter failure is assumed to cause the entire system to fail (if only

one voter is employed)

5.3 Reliability of a non-redundant system

We begin our analysis with the reliability of a non-redundant system. It is assumed that the basic module has passed through an extensive burn-in period , and the failure rate λ is constant, i.e. $R(t) = \exp(-\lambda t)$. In fact λ is a function of parameters such as temperature , vibration , humidity , ... , and of course time. Therefore it is possible to calculate the reliability of a system with (λ) as a function of the above parameters, but this is not in the scope of our discussion. The reliability of the system is shown in Fig . 5.1

5.4 - Reliability of a TMR System

As described in section 3.6.1 the first fault-tolerant design and possibly one of the most popular ones is the TMR structure which is formed by triplicating a single module and voting on the outputs of the three modules. Historically and practically TMR is an important redundancy technique to study. TMR serves as a benchmark against which other redundancy schemes are often compared [Ma69] [Ma70] [AbSi74]. Thus a thorough understanding of TMR reliability is necessary.

In addition to the general assumption made in the previous sections, the reliability of each individual module in a TMR structure is

assumed to be R_m, and failure of modules are independent of each other . The model that we use is the classical combinational (mixture of series and parallel) model.

R_{TMR} will be used for the system reliability with a TMR structure, and R_v for the reliability of the voter . Though for the time being we assume that the voting circuit does not fail, that is the reliability of the voter is 1. Hence the reliability of a TMR structure can be determined as a function of R_m as long as two modules are operating correctly.

Reliability of TMR = Reliability of all three modules working +

Reliability of all but one module working

$$R_{TMR} = R_m^3 + 3\,(1 - R_m)\,.\,R_m^2$$

$$R_{TMR} = \sum_{i=0}^{1} C^3_i (1 - R_m)^i . R_m^{(3-i)} \qquad 5.1$$

$$R_{TMR} = 3\,R_m^2 - 2\,R_m^3 \qquad 5.2$$

$$\text{if } R_m = e^{-\lambda t} \quad \text{then} \quad R_{TMR} = 3\,e^{-2\lambda t} - 2\,e^{-3\lambda t}$$

Fig 5.1 illustrates the reliability of a TMR system as a function of time and compares it with the reliability of its non-redundant module R_m. Reliability of a TMR system is better than that suggested by the above equation as the system may continue to operate properly even if two modules fail. For example if one of the modules is stuck-at-0 and another is stuck-at-1, the system still produces the correct output.

Fig. 5.1 - a The reliability of a non-redundant system and a TMR system as a function of time , are shown for the period t = 0 to t = 50000 hours.

The failure rate of the basic module is $\lambda = 6 * 10^{-5}$ failures per hours

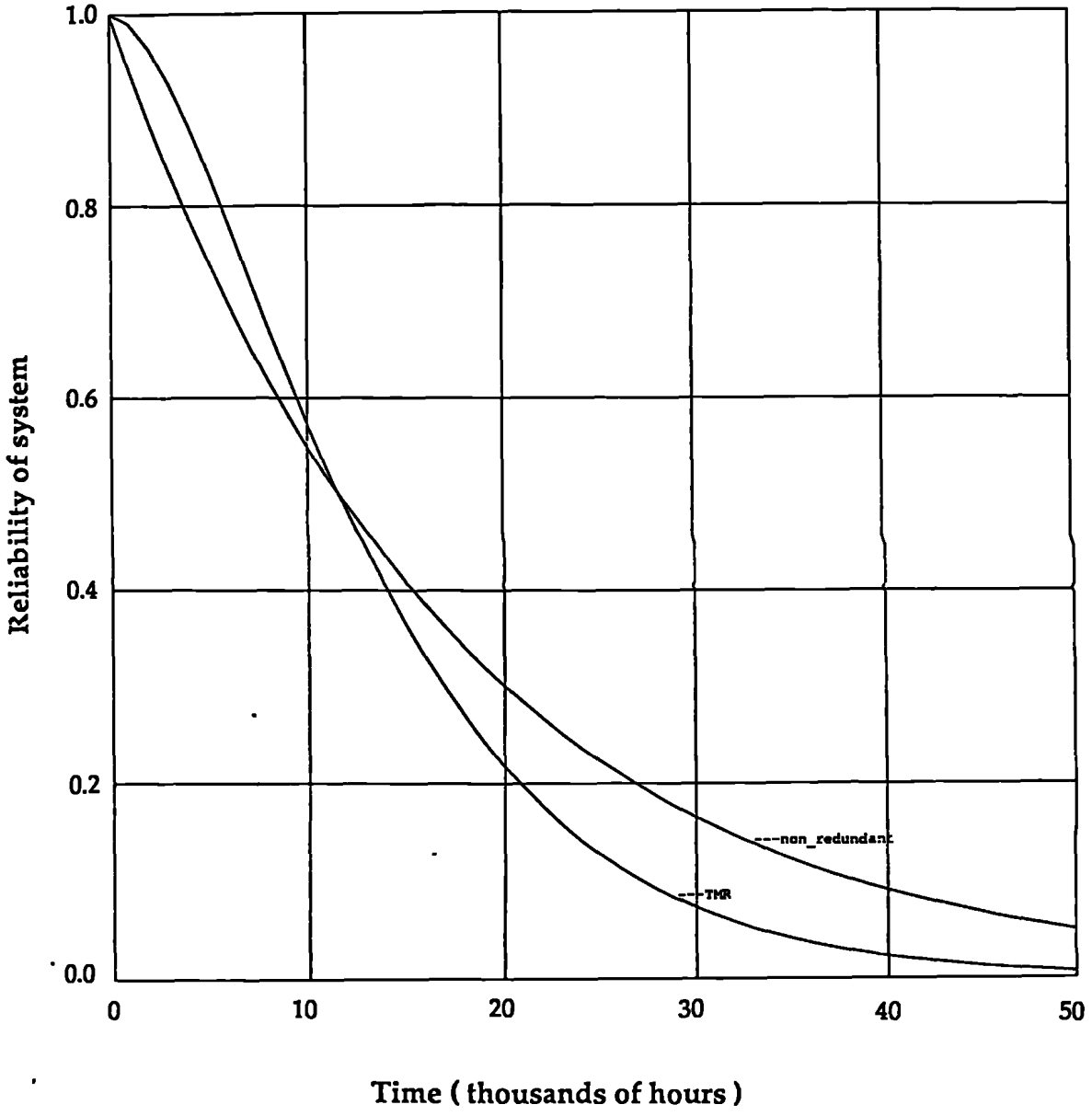
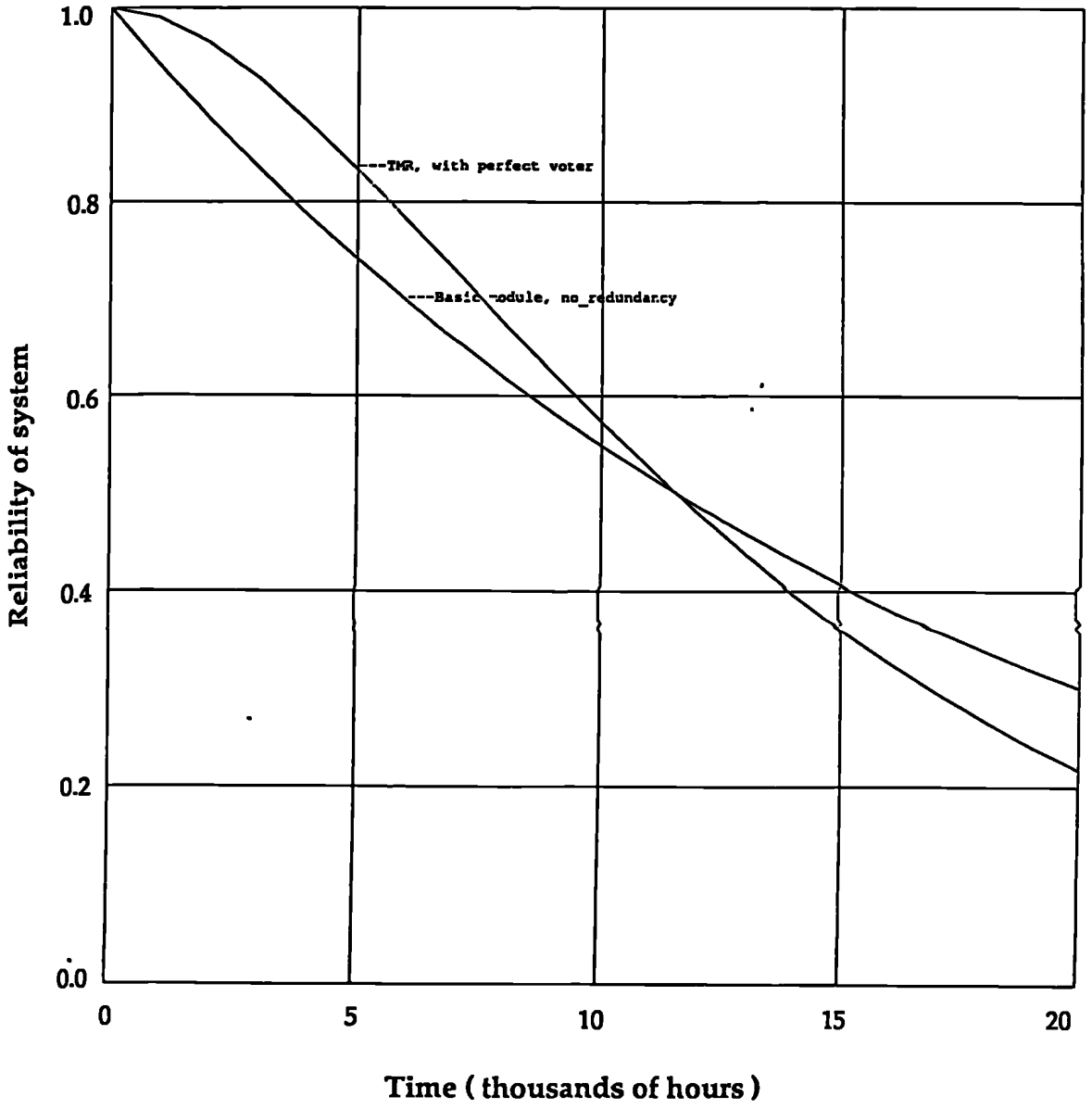


Fig. 5.1 - b The reliability of a non-redundant system and a TMR system shown for the period t = 0 to t = 20000 hours.
The failure rate of the basic module is $\lambda = 6 * 10^{-5}$ failures per hours



So far we have assumed that the voter is perfect. Now if we consider the voter reliability as (Rv), the system reliability will be

$$R_{TMR} = R_V(3R_m^2 - 2R_m^3) \qquad 5.3$$

The reliability of a system with imperfect voter is shown in Fig. 5.2
As mentioned earlier the system will fail whenever the voter fails. This is one of the important disadvantages of TMR. The solution to this problem is to triplicate the voters (as well as the modules) up to the last stage. At the last stage we have to accept this setback and therefore use a highly reliable voter [AbSi74]. Fig. 5.3 illustrates this structure.

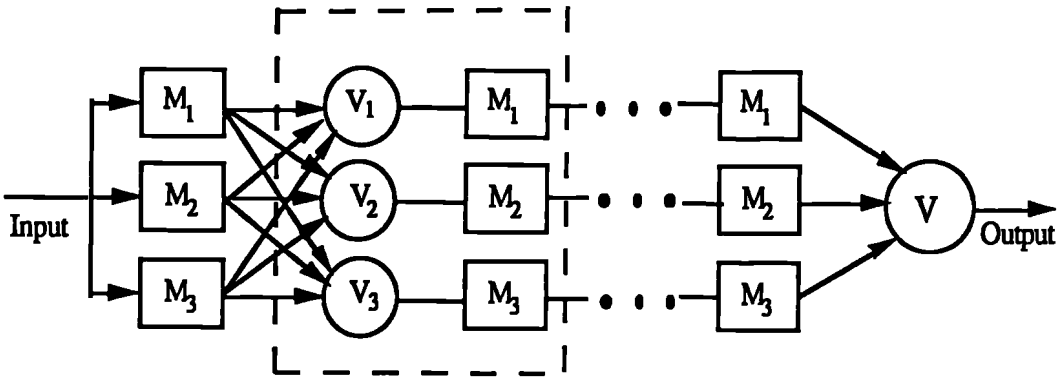


Fig. 5.3 The TMR structure is applied to voters as well as the modules.

$$R_{cell} = (R_m R_v)^3 + 3(R_m R_v)^2(1 - R_m R_v)$$

Using equation 5.3 for the reliability of a TMR (basic) the overall system reliability can be obtained by

$$R_{sys} = [R_{TMR}(\text{basic})] . [R_{cell}]^n$$

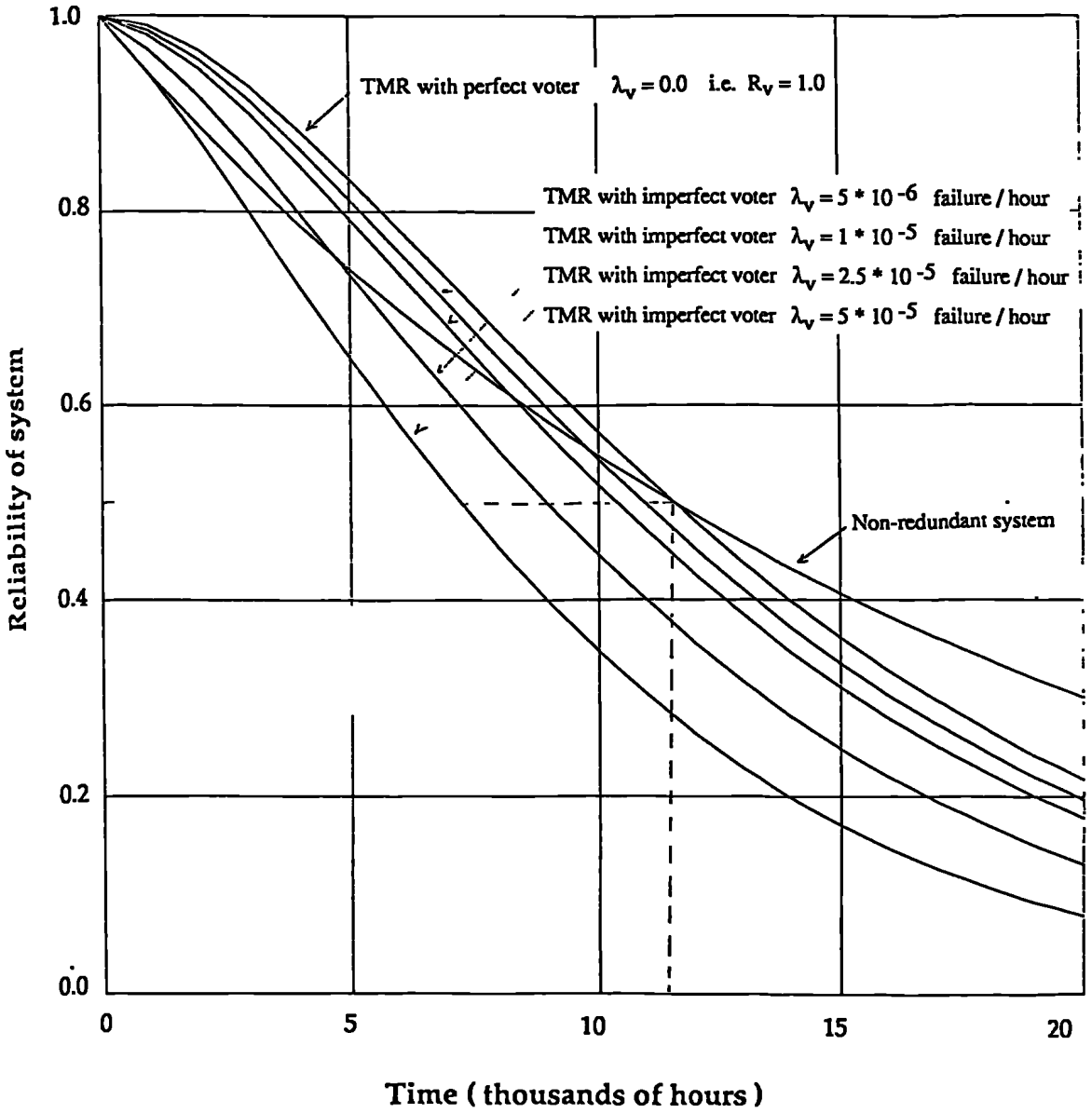
where n is the number of cells used in the system

$R_{cell} > R(\text{basic TMR})$ if $2 R_m R_v \geq 3 - 2 R_m$

Fig . 5.2 The reliability of a TMR system as a function of time , using different imperfect voters. Note that the reliability of the system will not be improved if R_v is less than 0.89 .

Time period is from t=0 to t= 20000 hours

The failure rate of the basic module is $\lambda = 6 * 10^{-5}$ failures per hours



Another problem with the TMR is if a module fails in a TMR system, both the remaining modules must continue to operate correctly to ensure that the system will operate correctly, so after the first failure the reliability of the system is less than that of an individual module.

e.g. after the first failure

$R_{TMR} \longrightarrow R_m^2$ and $R_m^2 < R_m$ because $0 < R_m < 1$

therefore the reliability will be improved if one of the working modules is switched off after a fault occurs. A system that consists of a TMR with a swicth-off mechanism is called TMR / simplex [BaHa69]. The overall system reliability of a TMR/simplex for a mission time T is

$$R_{sim}(T) = R_m^3(T) + \frac{3}{2}\left|R_m(T) - R_m^3(T)\right|$$

$(0 <= t <= T)$

The above equation is obtained as follows:

The probability that all the modules will survive the mission time T is $(R_m)^3$. If one of the modules fails at some time t and that module is removed together with one of the remaining good modules and the system (with only one of the fault-free modules) operates correctly for the rest of the mission time (T-t), then the probability of this event happens is

$$3\int_0^T [\{\frac{d}{dt}(1-e^{-\lambda t})\}\, e^{-2\lambda t}\, .\, e^{-\lambda(T-t)}]\, dt = \frac{3}{2}\left|R_m - R_m^3\right|$$

Therefore

$$R_{simplex} = R_m^3 + 1.5\,R_m - 1.5\,R_m^3 = 1.5\,R_m - 0.5\,R_m^3$$

The last equation shows that the reliability will be increased by switching off one of the good modules after occurrence of the first failure.

N.B. We can switch off the voter and make a direct connection, e.g. by forcing one of the good modules to be stuck-at-1 and the faulty module to be stuck-at-0.

Two major points should be noticed by the designer of a TMR system .

1) If $R_m < 0.5$ then $R_{TMR} < R_m$ even with a perfect voter i.e. $R_v = 1$

2) If $R_v < 0.89$ then $R_{TMR} < R_m$ irrespective of the value of Rm.

Fig. 5.2 illustrates these points.

The above values may be obtained mathematically as follows:

for the TMR structure should to increase the system reliability, we need $R_{sys} > R_m$, that is

$$(3R_m^2 - 2R_m^3)\, R_V > R_m \qquad 5.4$$

Assume that the voter is perfect i.e. Rv = 1 . Thus equation 5.4 can be written as follows:

$$3R_m^2 - 2R_m^3 - R_m > 0 \quad \text{or} \quad -2R_m(R_m^2 - \frac{3}{2}R_m + \frac{1}{2}) > 0$$

$$-2R_m(R_m - \frac{1}{2})(R_m - 1) > 0$$

That is for values of $R_m = 0$, $R_m = 1/2$, or $R_m = 1$ the term on the left side of inequality sign (denoted by R*) is zero. The solution to the above inequality is shown in the following table.

R_m	0		1/2		1
R*	0	—	0	+	0
			solution		

That is the solution is acceptable if $1 > Rm > 1/2$.

The assumption Rv = 1 is not realistic, therefore with an imperfect voter the inequality 5.4 may be solved as follows;

$$3R_m^2 - 2R_m^3 > \frac{R_m}{R_v}$$

$$3R_m - 2R_m^2 - \frac{1}{R_v} > 0$$

This is a quadratic equation and can be rearranged as follows:

$$R_m^2 - \frac{3}{2}R_m + \frac{1}{2R_v} < 0 \quad \text{or} \quad \left(R_m - \frac{3}{4}\right)^2 - \frac{9}{16} + \frac{1}{2R_v} < 0$$

$$\text{or} \quad \left(R_m - \frac{3}{4}\right)^2 < \frac{9R_v - 8}{16R_v}$$

In order to have solution to the above inequality (in terms of real numbers) the term ($9 R_v - 8$) / R_v must be either positive or zero. i.e.

$$9 - \frac{8}{R_v} >= 0 \quad \text{, therefore } R_v >= \frac{8}{9} \quad \text{or } R_v >= 0.89$$

For Rv = 0.89 , the minimum value for Rm is 0.75 .

Thus the requirements for a TMR to be useful are:

1- With a perfect voter the minimum value of module reliability is 0.5

2- The minimum value for the reliability of a voter to be used in a TMR structure is 0.89 , then with such a voter the minimum reliability of a module to be used is 0.75 .

Considering the reliability of the TMR system evaluated above and the curves plotted in Fig. 5.1, one concludes that TMR technique is not suitable for long missions, as the system reliability will be even less than a non-redundant system if the mission time is long. An important point in the TMR scheme is that particular attention should be paid to the reliability of the voter.

The important role of the voter in producing error-free computation is obvious. Failure in the voter used in the system , amounts to system failure.

The majority voter for a TMR is very simple and its logic diagram is shown in Fig. 5.4

$$Z = X_1X_2 + X_1X_3 + X_2X_3$$

Using De Morgan's Theorem we obtain:

$$Z = \overline{\overline{X_1X_2}.\overline{X_1X_3}.\overline{X_2X_3}}$$

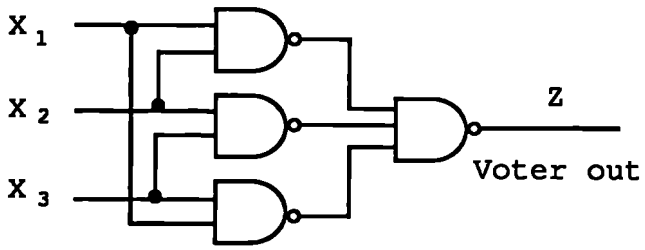


Fig. 5.4 Logic diagram of a TMR voter

Reliability of this voter can be obtained as a function of reliability of the gates used in the circuit.

$$R_v = R_{NAND}^4 \qquad 5.5$$

For example, for a Bipolar TTL NAND gate estimated failure rate is 0.011 per million hours (field failure rate is 0.015 per million hours). If we assume the probability function for this gate is exponential, then

$$R_v = \exp(-\lambda_{NAND} . t)^4 = \exp(-\frac{0.015}{10^6} . t)^4$$

Reliability of the voter is normally more than that shown in the equation 5.5 . For example, if one of the NAND gate's output is stuck-at-0 the correct output still may be obtained. The reliability of each gate depends on the technology used and also on the realisation of the gate itself.

5.5 - Reliability of an NMR Structure

NMR is the general form of a TMR for any odd number N [MaSo75]. If n=(N-1)/2, then an NMR system can tolerate as many as n module failures. The reliability of such a system is , therefore

$$R_{NMR} = [\sum_{i=0}^{n} C_i^N (1 - R_m)^i . R_m^{(N-i)}] . R_v$$

where C_i^N is combinatorial N object taken i at a time

Higher reliability is expected as the number of modules in an NMR structure increases. But on the other hand the complexity of the voter increases non-linearly with the number of modules. Thus there is an optimum number of modules that results in maximum reliability occurrence, using the NMR technique. Figures 5.5 and 5.6 show the reliability of a TMR, a 5MR, and a 7MR system with perfect and imperfect voters respectively.

Fig. 5.5 shows the reliability of a TMR, a 5MR, and a 7MR system as a function of time, and compare them with a non-redundant system. The voter used in each system is assumed to be perfect.

Time period is from t=0 to t=20000 hours .

The failure rate of the basic module is $\lambda = 6 * 10^{-5}$ failures per hours

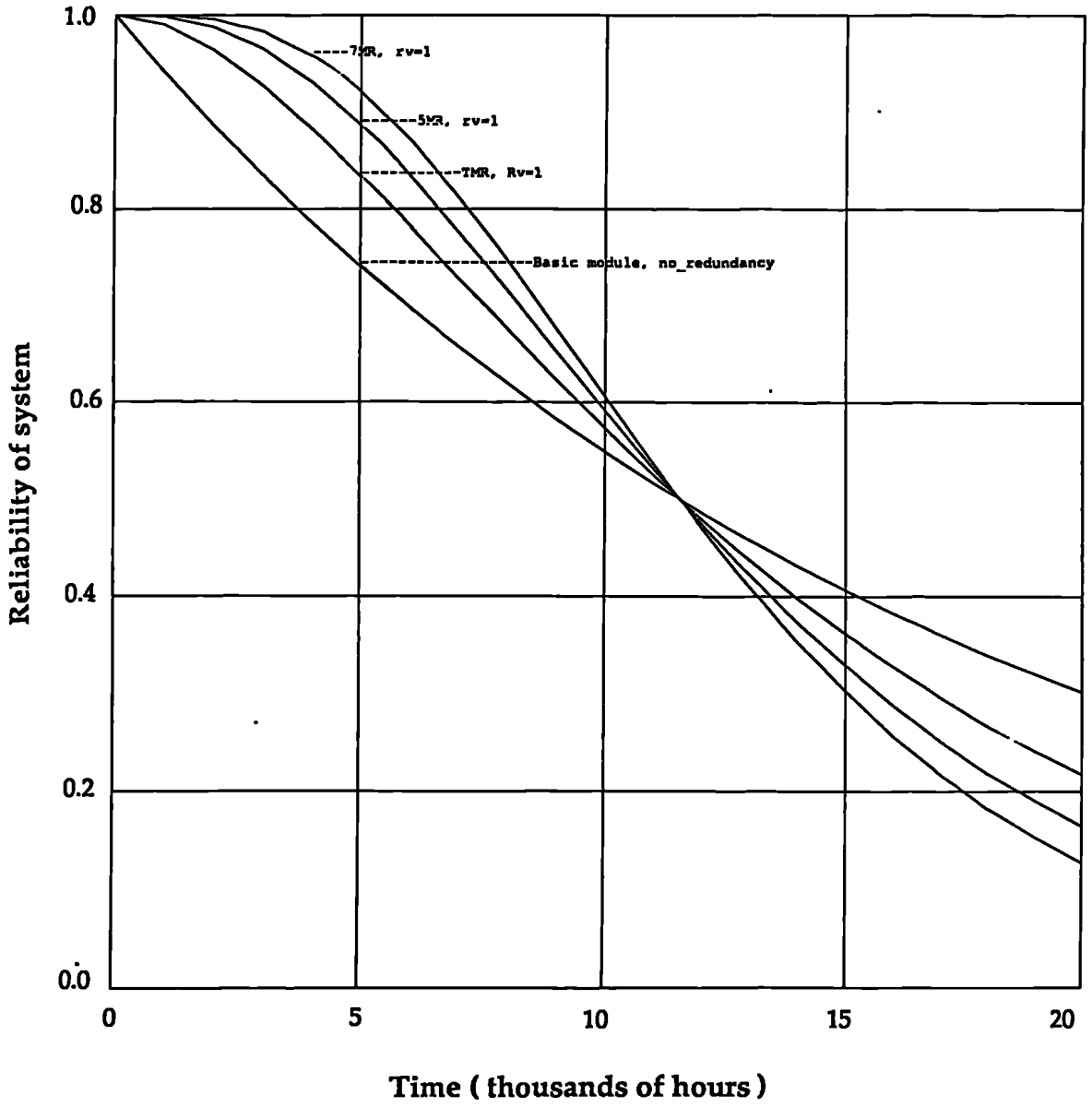
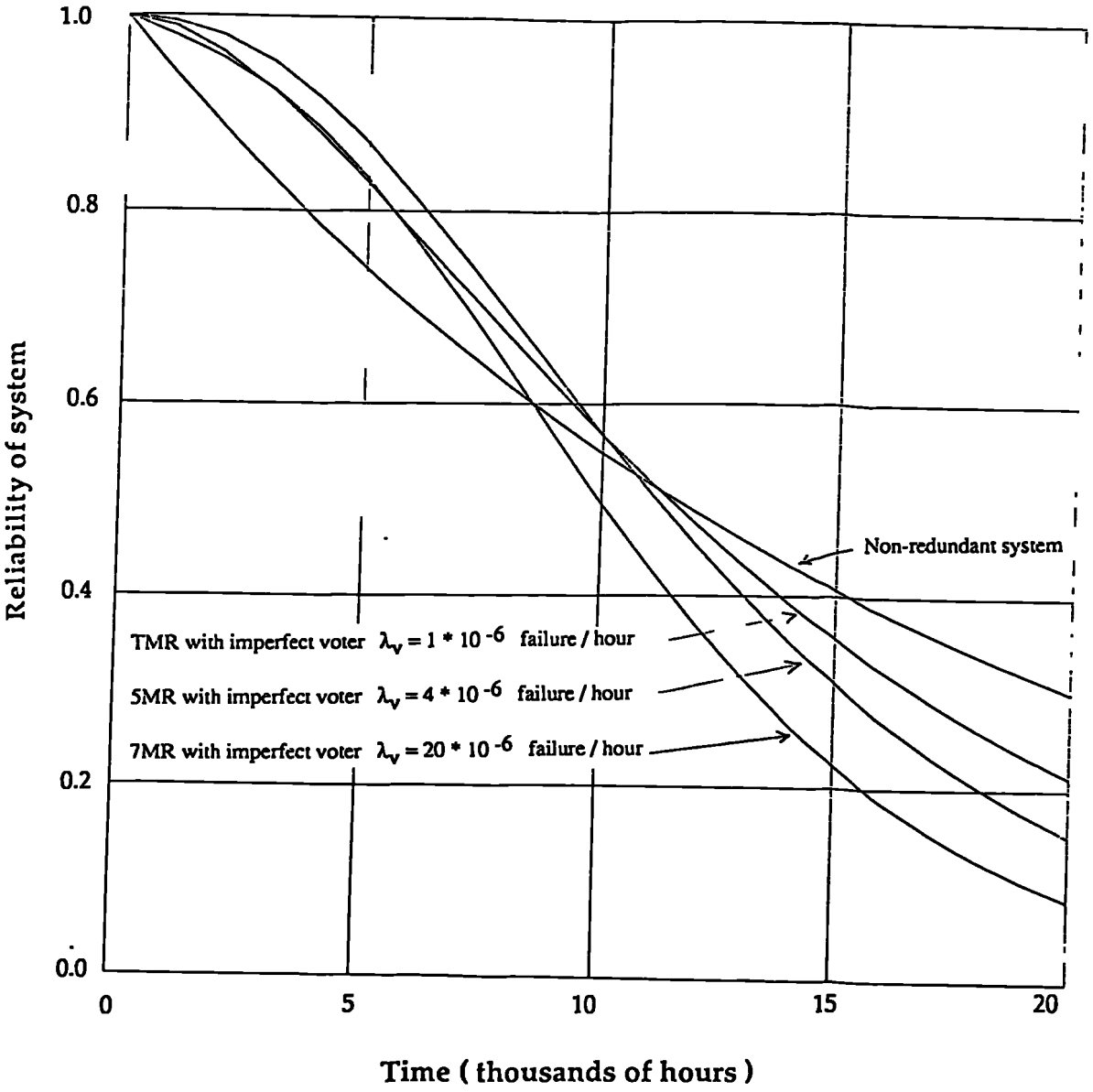


Fig. 5.6 shows the reliability of a TMR, a 5MR, and a 7MR system as a function of time with imperfect voters, and compare them with a non-redundant system.

Time period is from t=0 to t=20000 hours .

The failure rate of the basic module is $\lambda = 6 * 10^{-5}$ failures per hours



5.6 - Reliability of dynamic redundancy structure.

The dynamic redundancy technique has been described in chapter three. In this type of design there is one active module and one or more spares. The switching circuit will disconnect the active module and replace it with a spare module if a failure is detected in the active module. It is assumed that the modules are independent and the reliability of each module is R_m. The system is working properly as long as only one fault-free module remains. Thus the system reliability can be expressed by equation 5.6.

$$R_{sys} = [1 - (1 - R_m)^{s+1}] R_s . R_d \qquad 5.6$$

Rs is the switch reliability

Rd is the detector reliability and

$R_{sys} = 1 - (1 - R_m)^{s+1}$ with perfect switch and detection circuit

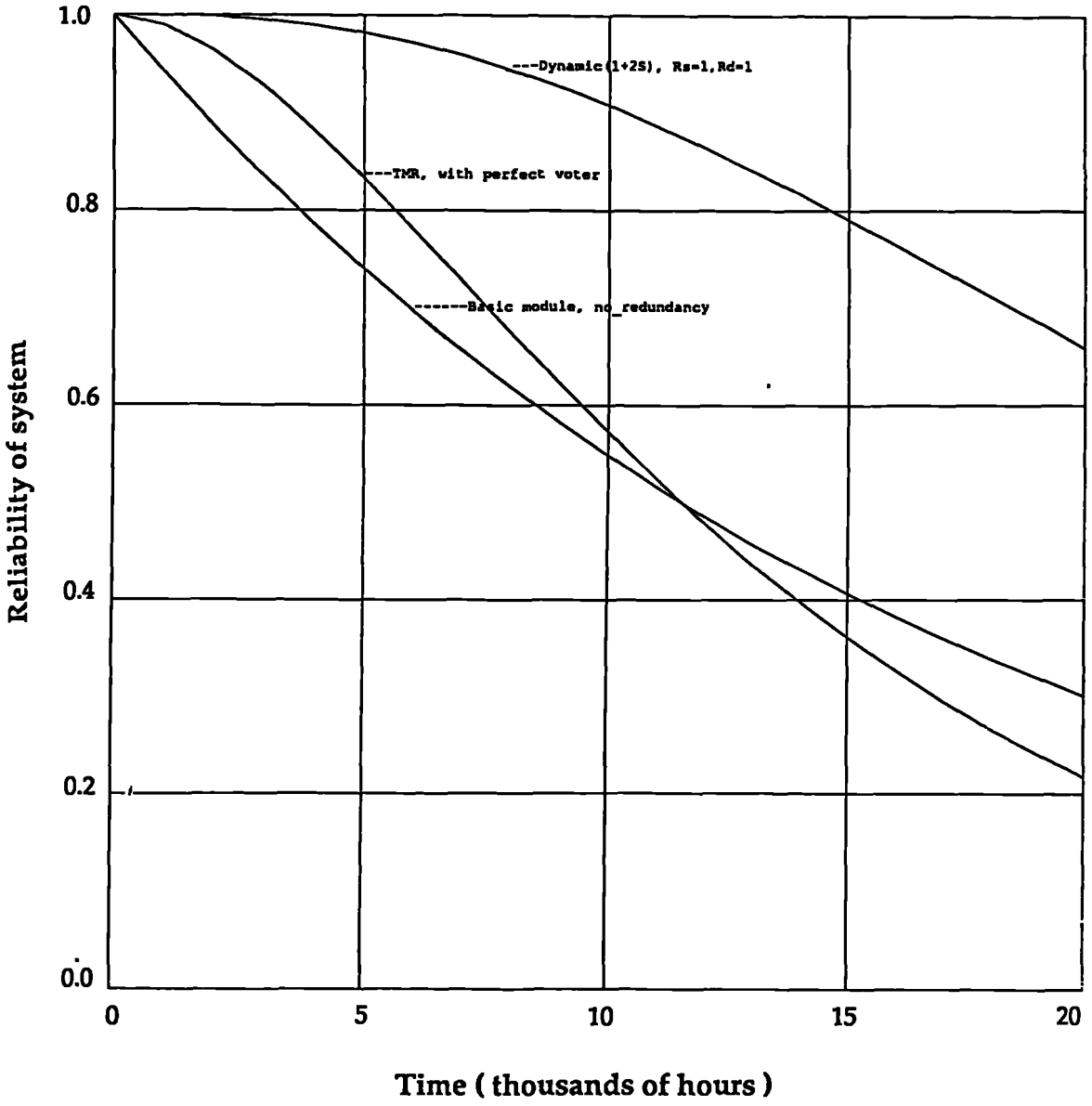
In the equation 5.6 the importance of a reliable switch and detection circuit is obvious . Diagram 5.7 shows the reliability of a system with one active module and two spares , and compares it with the reliability of a non-redundant system. In this structure the switch and the detection circuit become very complex when many spare units are used , thus the reliability of the system decreases significantly, resulting in an inefficient system.

There are many practical systems using one active unit and one spare. For example, the Bell Electronic Switching System ESS [To78] .

Only the cost and application of systems can determine which technique should be used, the TMR, the dynamic with 1 spare , or the dynamic with more spares, if high reliability is not the only criteria.

Fig. 5.7 illustrates the reliability of a dynamic system (as a function of time) consisting of one active module and two spares with a perfect switch and disagreement detection circuit. The improvement over a non-redundant and a TMR system is shown.

The failure rate of the basic module is $\lambda = 6 * 10^{-5}$ failures per hours



5.7 - Reliability of hybrid designs

Hybrid structure consists of an NMR core, some spare modules, a disagreement detection circuit, a switch, and a voter. The system output of a hybrid structure with N modules in its core is correct if (N - 1) / 2 modules operate properly, provided that the switch , the failure detection circuit, and the voter are fault-free. So the reliability of a hybrid structure with a TMR core and S spares can be expressed by the following equation

$R_H(3, S) = 1 - \{$ probability of all $(S + 3)$ modules failing + probability of all but one module failing $\}$

$$= 1 - \{(1 - R_m)^{(S+3)} + (S + 3) . R_m . (1 - R_m)^{(S+2)}$$

$$= 1 - (1 - R_m)^{(S+2)} \{ 1 - R_m + S . R_m + 3 R_m \}$$

$$= 1 - (1 - R_m)^{(S+2)} \{ 1 + R_m (S + 2) \}$$

In general the reliability of a Hybrid (N,S) system is

$$R_{H(N,S)} = \sum_{i=0}^{n+S} C_i^{N+S} . (1 - R_m)^i . R_m^{(N+S-i)} \qquad 5.7$$

where: S = number of spares and

N = number of modules in the core

n = (N - 1) / 2 fault-tolerance of the core

$$R_{H(3,2)} = \sum_{i=0}^{3} C_i^5 . (1 - R_m)^i . R_m^{(5-i)}$$

$$= R_m^5 + 5(1 - R_m)^1 R_m^4 + 10(1 - R_m)^2 R_m^3 + 10(1 - R_m)^3 R_m^2$$

The above equation expressed the system reliability with a perfect Voter, Switch, and Disagreement detection circuit (VSD). If the reliability of these circuits (i.e. R_V, R_S, and R_d) were taken into account, the system reliability would be

$$R_{sys}(3,2) = [\, R_m^5 + 5(1 - R_m) R_m^4 + 10(1 - R_m)^2 R_m^3 + 10(1 - R_m)^3 R_m^2 \,] R_v R_s R_d$$

The important role of the VSD reliability in this scheme in determining the overall system reliability [Og74] [Og75] should be noted by the system designers.

A reliable switch design for a hybrid (3, 2) system has been proposed by Siewiarek [SiMc73]. An example of using this type of redundancy is the Self Testing And Repairing computer STAR [AvEt71].

The reliabilities of hybrid (3, 1) and hybrid (3, 2) are shown in Fig. 5.8 The switching and the voting mechanisms are assumed to be fault-free. The reliability of a non-redundant and a TMR system are also plotted

on the same graph for comparison. Figure 5.9 illustrates the reliability of systems with imperfect VSD circuits . Several values have been chosen for Rv, Rs, Rd.

Generally Rv > Rd > Rs , because the switch is more complex than the detection circuit , and the detection circuit is more complex than the voter.

In practice the system reliability of hybrid is higher than the value shown in equation 5.7 because the failure rate of the passive spare modules (μ) is less than the failure rate of the active modules (λ), in the core.

The reliability of a system using hybrid structure is greater than the reliability of a system using TMR, or a system using its equivalent NMR structure . But for long mission time, the reliability of all of these redundant designs will be less than the reliability of a single module, as is shown in the graphs. So the immediate result is that a system with no redundancy is more reliable than a system with redundancy in very long missions. But there are other parameters to be considered, because faults will occur during the operation of the system. The first one is the *fault tolerance* , that is the number of faults that can be tolerated by a system. If we have a non-redundant system, the fault tolerance of the system is zero, i.e. after the occurrence of the first failure we get the wrong output and if that system is used for an important task the

consequence could be catastrophic . But if we use a fault-tolerant system, one or more faults can be tolerated without affecting the final output.

Another parameters is *availability* . There are cases that nonstop operation of a system is required even with the presence of faults. This requirement can not be achieved by a non-redundant system (e.g. telephone switching systems).

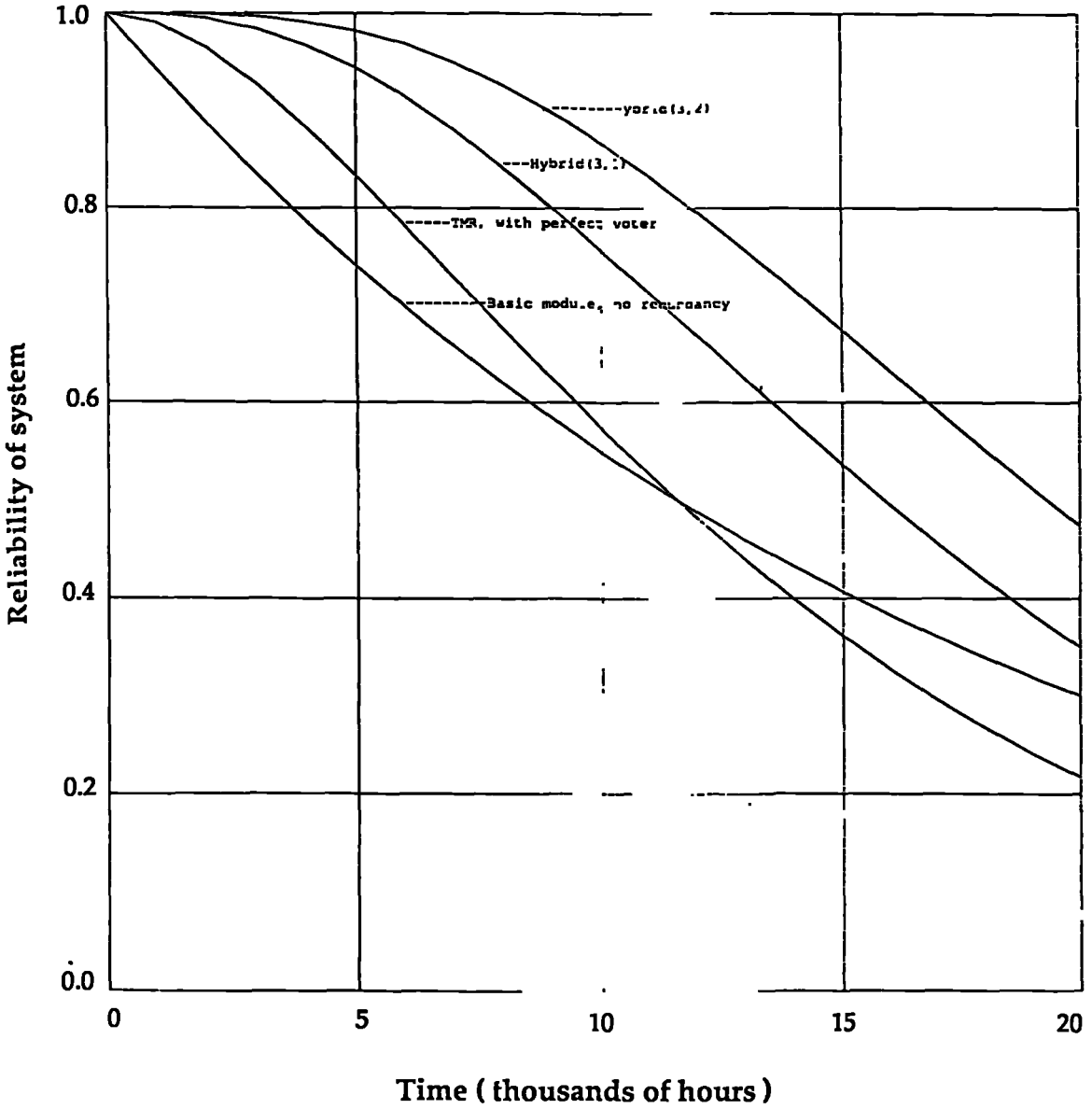
Systems used in *Long-life applications* (for instance in spacecraft where systems require long periods of unattended operation) must also have fault-tolerant mechanisms.

Critical application is another example in which a system is designed to undertake critical tasks therefore it must have the ability to tolerate permanent and transient faults which is not possible using a non-redundant system.

The above discussion shows that fault-tolerance is necessary in many digital system designs even if highly reliable computation is not required.

Fig 5.8 shows the reliability of a hybrid system with a TMR core and 1 spare, and a hybrid system with a TMR core and two spares as a function of time, (t=0 to t=20000). Voter and reconfiguration mechanism are assumed to be perfect.

The failure rate of the basic module is $\lambda = 6 * 10^{-5}$ failures per hours



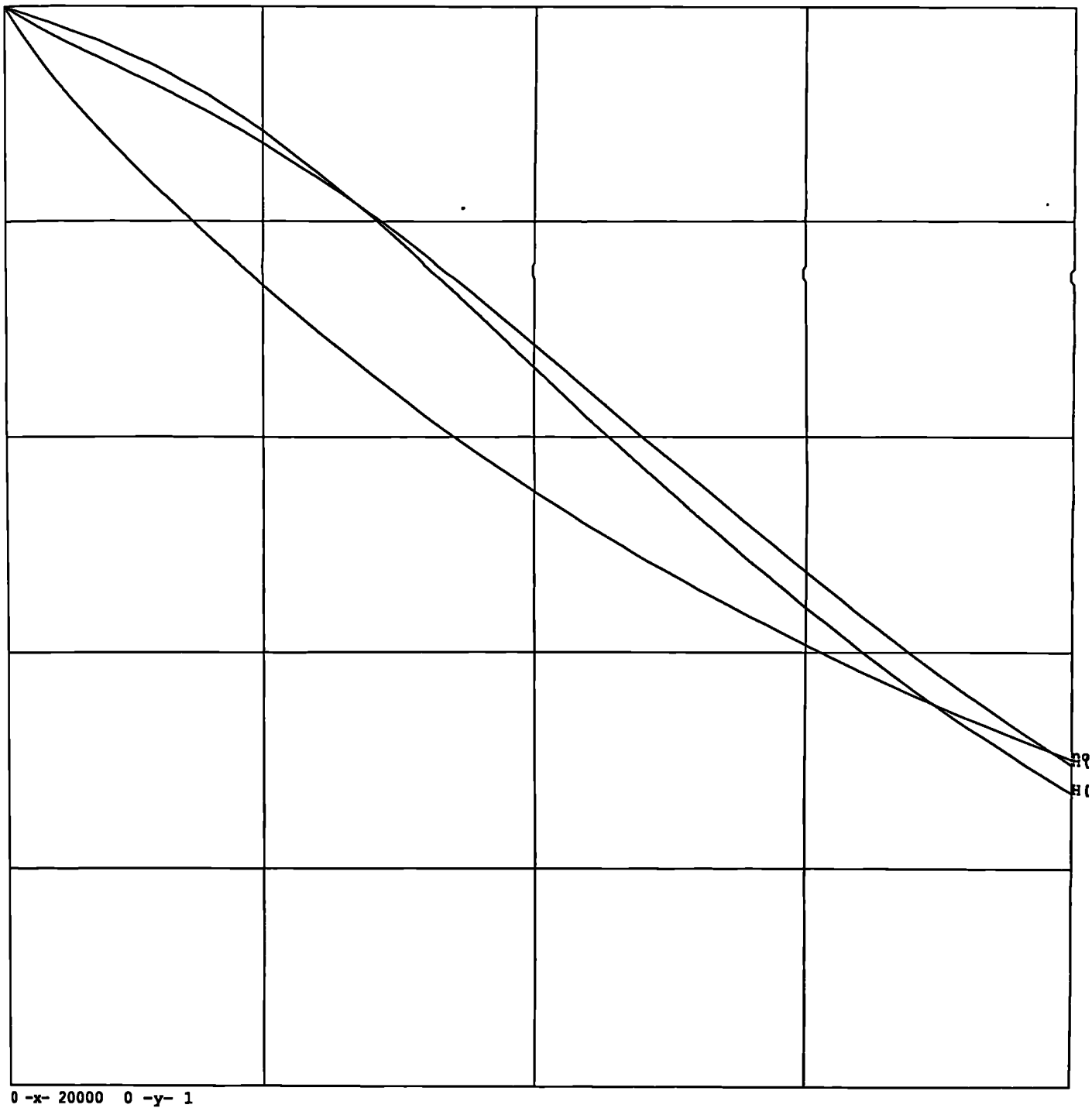
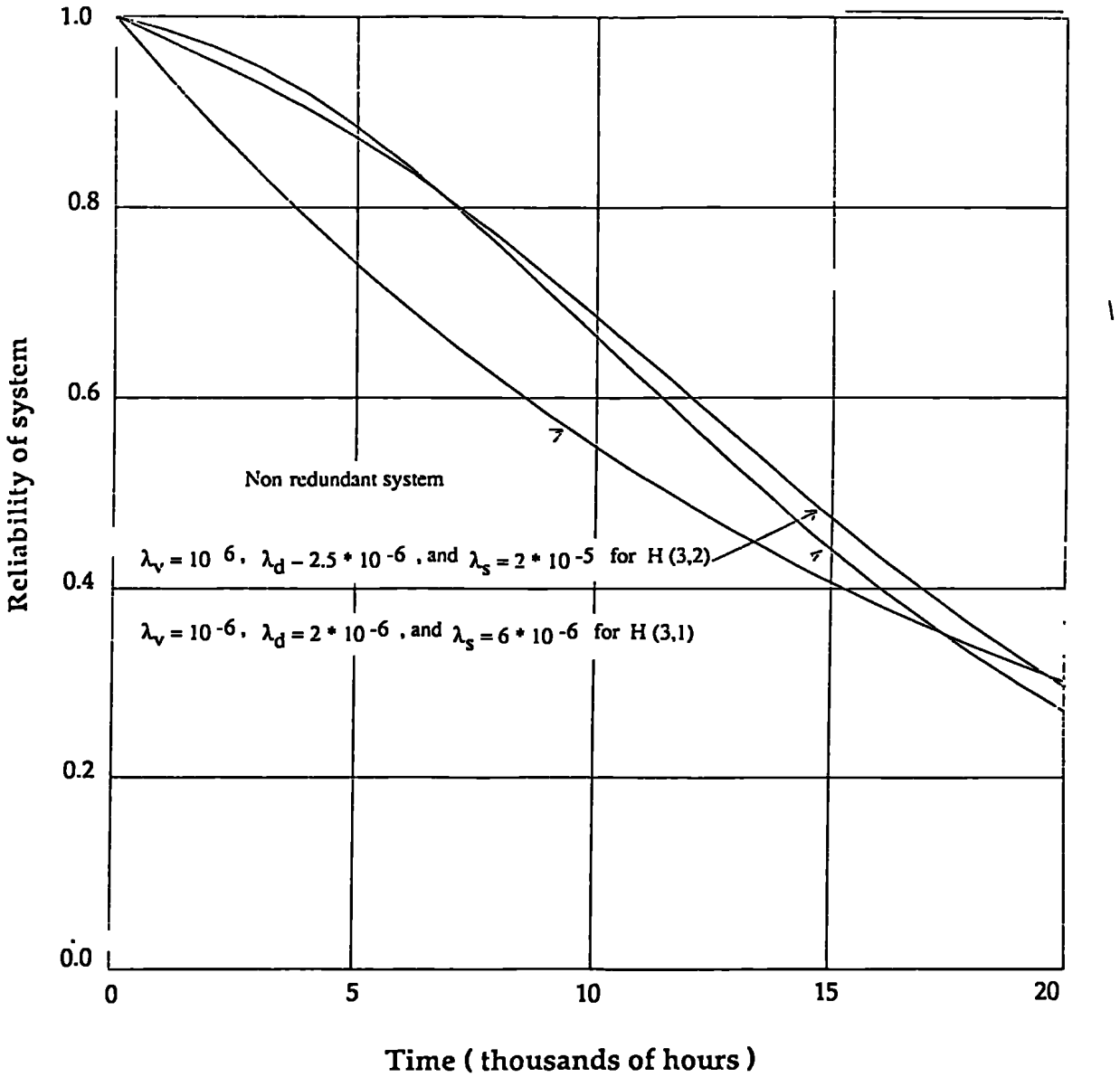

0 -x- 20000 0 -y- 1

Fig. 5.9 shows the reliability of a hybrid system with a TMR core and 1 spare, and a hybrid system with a TMR core and two spares as a function of time, (t=0 to t=20000). Voter and reconfiguration mechanism are not perfect (their reliabilities are shown in the graph).

The failure rate of the basic module is $\lambda = 6 * 10^{-5}$ failures per hours



We have evaluated the reliability of the systems using the classical model, and compared the overall system reliability of the most common fault-tolerant configurations. Frequently the classical model does not give accurate reliability calculations . The reason is often the generality of the model. Therefore in the following sections, a model that is appropriate for the reliability evaluation of the HR-HE and the MFT-RS schemes will be developed.

5.8 - Levels of Reliability Models

Typically, a reliability model divides a structure into various subsections that are easier to study than the whole structure itself. The technique to be applied depends upon the amount of detail given about each subsection. The detail in each subsection is itself a function of the desired accuracy of the model. We will now describe levels at which systems may be modelled. It is to be noted that often the boundary between the levels is not clear, and the distinction is often merely for convenience or understandability.

The highest level of modelling is that of the system level, in which an entire hardware system is treated as a black box. A large number of observations are made about events (e.g. , failures of a certain kind) , and the time intervals between events (i.e. time for recovery of the system) . A model may then be proposed to fit the data as closely as

possible . An enormous amount of observations are needed for successful modelling . The problem is especially hard to tackle if there are many types of events .

The next level at which a model may be attempted is the module level . The system is divided into a number of modules which have mutually independent failure probability distributions . The model of the system is obtained by a composition of the models of the modules . If it is not possible to divide a system into independent modules , the modules are so formed that they have nearly independent failure distributions . An approximate model is arrived at by assuming the modules to be independent . The modules themselves need to be modelled individually if they are not identical . Classical hardware reliability modelling has occurred at this level .

To obtain a model for a hardware module one may have to go to a lower level . This is the gate level of modelling , and gate reliability is often the basic parameter used to obtain the system reliability . Almost always the gates are assumed to have independent probability distributions in the absence of redundancy . Although this assumption is sometimes not realistic, any diversion from it substantially complicates the mathematics . An exponential distribution is commonly assumed for the gate reliability . Under the assumption of independence, if a gate reliability is p , and a module with

non-redundant logic has m gates , the module reliability is P^m .

In most systems , one does not have to go below the gate level . However , if the redundancy is introduced at a lower level , we need the component level of modelling , where components are transistors , diodes , resistors , etc . We will have occasion to use this level at a later stage (for example for the voters used in the fault-tolerant designs which will be discussed in a later section).

There are several reasons why the module level of reliability modelling should be studied . A first-order approximation of the reliability of large hardware systems can be easily derived by assuming module independence and counting components in each module . The number of parameters are usually few and dominating parameters or architectural features are identified . Further , various architectures can be compared on a gross scale . For these reasons , in the next section the emphasis is on the module level of reliability modelling .

5.9 - A new reliability model

In this section a new model is introduced to evaluate the reliability of HR-HE structure. An N-channel HR-HE structure may produce correct output as long as two out of N modules operating correctly. Therefore the overall system reliability may be calculated by the following

equation:

$$R_{sys} = [1-(1-R_m)^N - N(1-R_m)^{(N-1)}.R_m].R_{switch}.R_{voter}$$

In addition to the assumption made for the classical model discussed earlier , let us assume that the switch components in HR-HE are independent. We define R_s as the reliability of each switch component as illustrated in Fig. 4.9. Let us also introduce two new parameters, α and β which will be used to relate the reliability of the switch and the voter, (with three inputs) respectively , to the module reliability R_m . We will let $R_s = (R_m)^{\alpha}$ and $R_v = (R_m)^{C.\beta}$ where C is a factor which reflects how complex the voter is compared to the three input voter used by a three channel system. As it is assumed that the switch components are independent, each switch component may be included in the corresponding module , and the reliability of the modified module (defined as the cascade of the module with its switch component) can be computed . Then the overall system reliability may be calculated by

$$R_{sys} = [1-(1-R_mR_m^{\alpha})^N - N(1-R_mR_m^{\alpha})^{N-1}.R_mR_m^{\alpha}].R_m^{C.\beta}$$

Fig. 5.10-a shows the reliability of a three channel, a five channel , and a seven channel HR-HE structure with perfect switch , and voter, that

is $\alpha = 0.0$, and $\beta = 0.0$.

The overall system reliability of the same systems with imperfect switches and voters are shown in Fig. 5.10-b and Fig.10-c .

To calculate the reliability in this case, it is assumed that $\alpha = 0.1$, and $\beta = 0.02$. The value for α is chosen arbitrarily , because HR-HE may be used for any module , and choosing $\alpha = 0.1$ indicates that the module complexity of the employed module is ten times greater than the switch component complexity .

It should be noted that the switch and the voter realisations are known if a particular fault-tolerant scheme is selected , that is the number of gates or transistors used by the switch and the voter are known . Thus a relation may be established between the complexity of the voter and the switch component. For the above calculation a particular realisation is used for the voter (which will be discussed later) . Now if $\alpha = 0.1$ we may obtain a value to assign it to β . For example if a three channel HR-HE is used , $\beta = 0.02$. For a five channel $\beta = 0.03$, and for a seven channel $\beta = 0.05$.

Fig. 5.10 - a System reliability of (3, 5, 7 channels) HR-HE as a function of time, with perfect voter , switch, and disagreement detection circuit. Time period (t=0 to t=20000 hours).
The failure rate of the basic module is $\lambda = 6 * 10^{-5}$ failures per hours

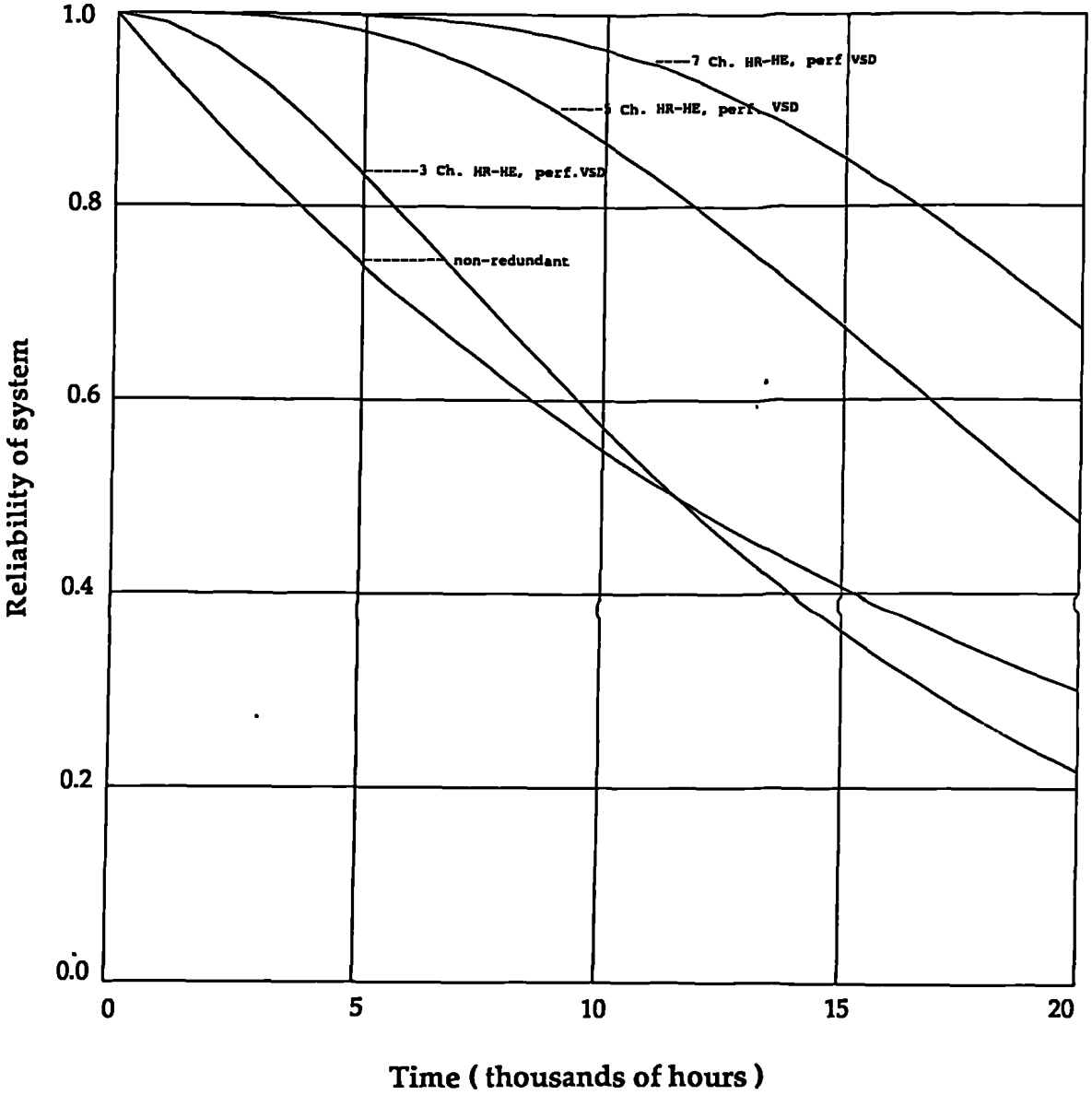
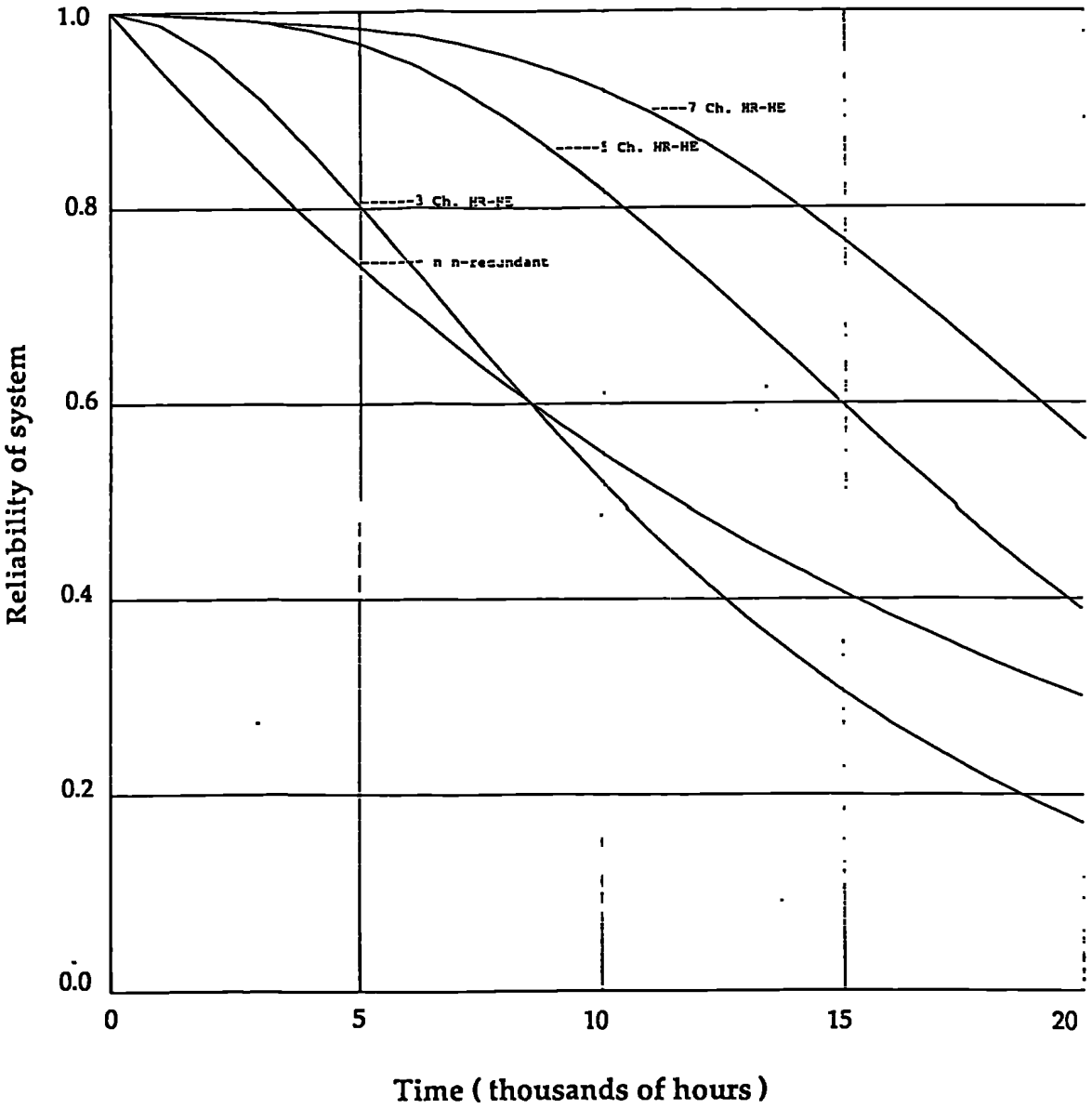


Fig. 5.10 - b System reliability of (3, 5, 7 channels) HR-HE as a function of time, with imperfect voter (b=0.02, 0.03, and 0.05 respectively), switch and disagreement detection circuit (a=0.1) as shown in the graph.

Time period (t=0 to t=20000 hours).

The failure rate of the basic module is $\lambda = 6 * 10^{-5}$ failures per hours

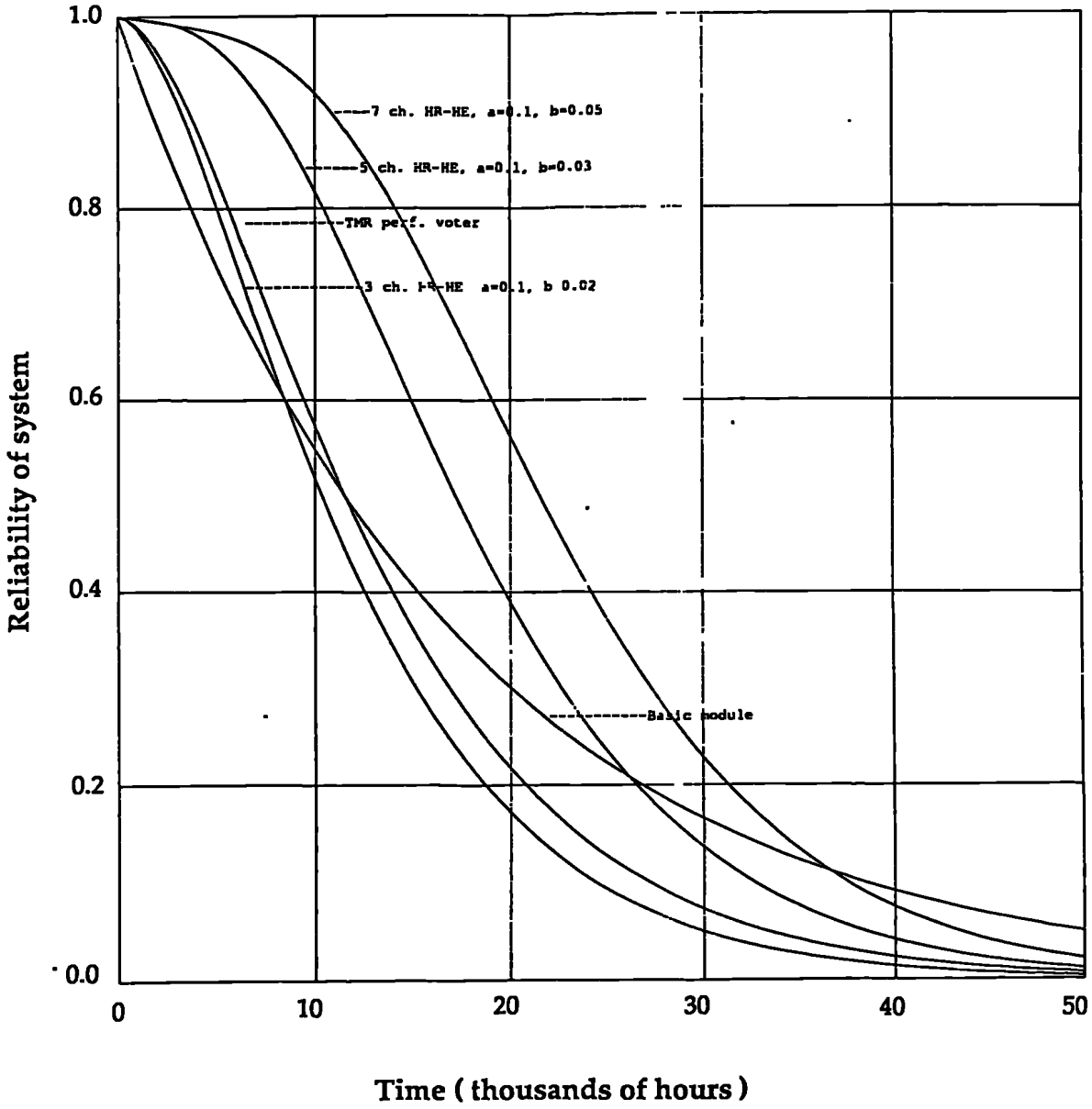


N.B. In figures 5.10 to 5.17 variables 'α' and 'β' are represented by 'a' and 'b'

Fig. 5.10 - c System reliability of (3, 5, 7 channels) HR-HE as a function of time, with imperfect voter (b=0.02, 0.03, and 0.05 respectively), switch and disagreement detection circuit (a=0.1) as shown in the graph.

Time period (t=0 to t=50000 hours).

The failure rate of the basic module is $\lambda = 6 * 10^{-5}$ failures per hours



5.10 - Comparing Schemes

There are a few measures for evaluation and comparison of different schemes . System reliability , R_{sys} , is the most obvious measure . As will be evident from the examples that will be presented , the graphs of R_{sys} against the number of modules (N) effectively high light the dependence of R_{sys} on (N). This is shown in Fig. 5.11.
However , for comparing systems that are highly reliable , R_{sys} is not a good measure of comparison.

Another absolute measure that may be used is the mission time . The mission time T_m is defined to be the time at which R_{sys} is exactly equal to some predetermined value (R_{sysmin}). It is , in other words , the time after which R_{sys} drops below that required for the minimum system reliability desired. T_m may be determined by using :

$R_{sys}(T_m) = R_{sys\,min}$.

A more interesting measure , and one that will be used in the examples that follow , is a comparative one , namely , the Mission Time Improvement (MTI) . MTI is defined to be the ratio of the mission times of the two schemes to be compared, which is a function of the R_{sysmin}. It is a more useful measure because MTI may be determined more easily. If for schemes 1 and 2 $R_{sys1}(t_1) = R_{sys2}(t_2)$,

then the MTI of scheme 1 over scheme 2 is (t_1 / t_2), where t_1 and t_2 are the mission times of system$_1$ and system$_2$ respectively. If the two schemes have modules with identical failure rates λ , with modules reliability $R_1 = \exp(-\lambda t_1)$ and $R_2 = \exp(-\lambda t_2)$, the MTI equals ($\text{Ln}\, R_1$ / $\text{Ln}\, R_2$) .

MTI of HR-HE scheme over TMR:

The MTI of the HR-HE scheme over TMR was found in the following manner. Assuming identical modules (having the same failure rate λ), a value for R_1 , the reliability of a module in the HR-HE was picked arbitrarily. The equation $R_{HR\text{-}HE}(R_1) = R_{TMR}(R_2)$ was solved for R_2 . Since $R_1 = \exp(-\lambda t_1)$ and $R_2 = \exp(-\lambda t_2)$, $\text{Ln}\, R_1$ / $\text{Ln}\, R_2$ yields the desired MTI. When MTI > 1 , the HR-HE has longer Tm than TMR scheme. Results of this computation are presented in Fig. 5.12.

Fig. 5.11 The reliability of HR-HE as a function of the number of modules for three different values of module reliability.
The failure rate of the basic module is $\lambda = 6 * 10^{-5}$ failures per hours

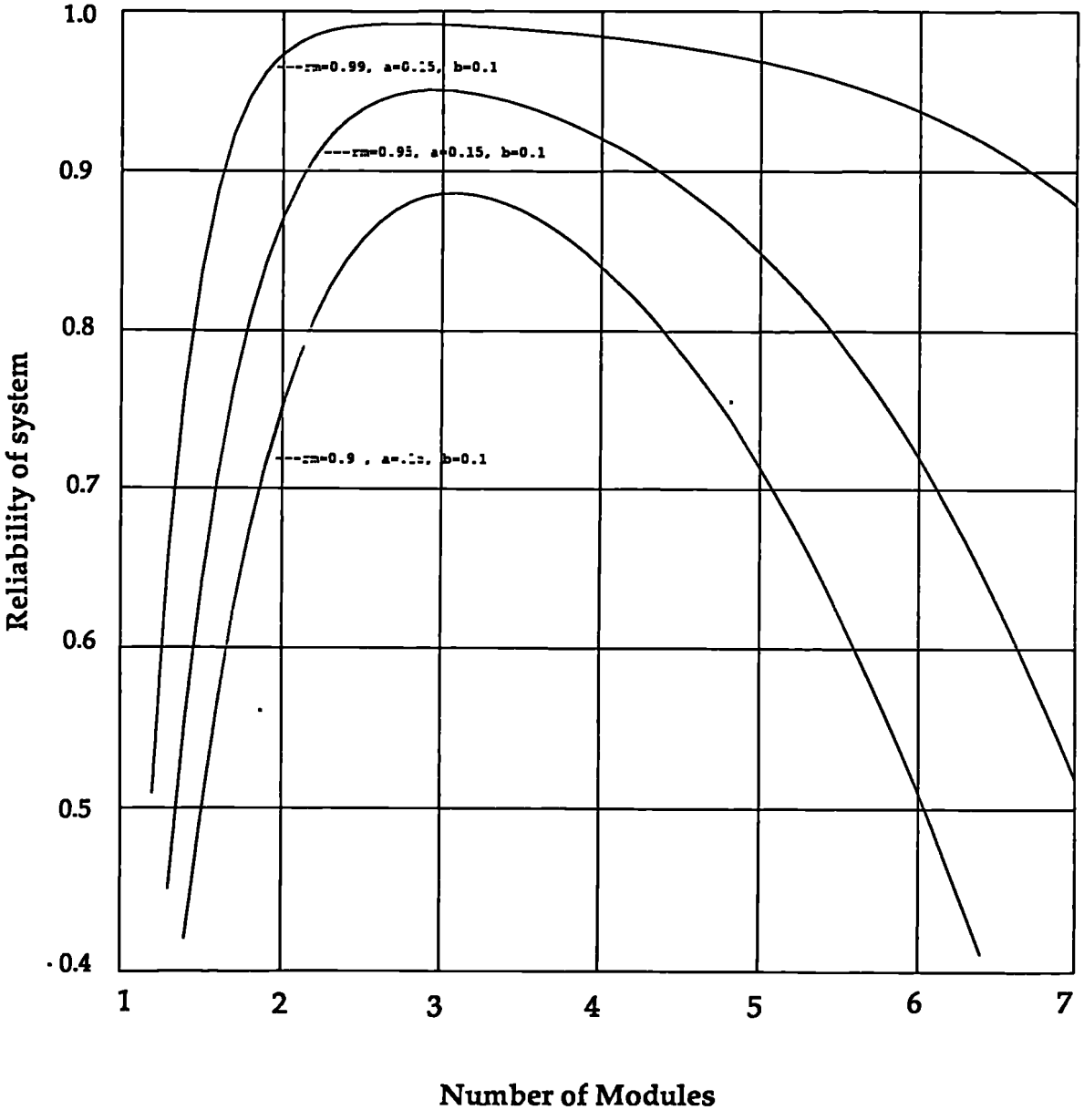
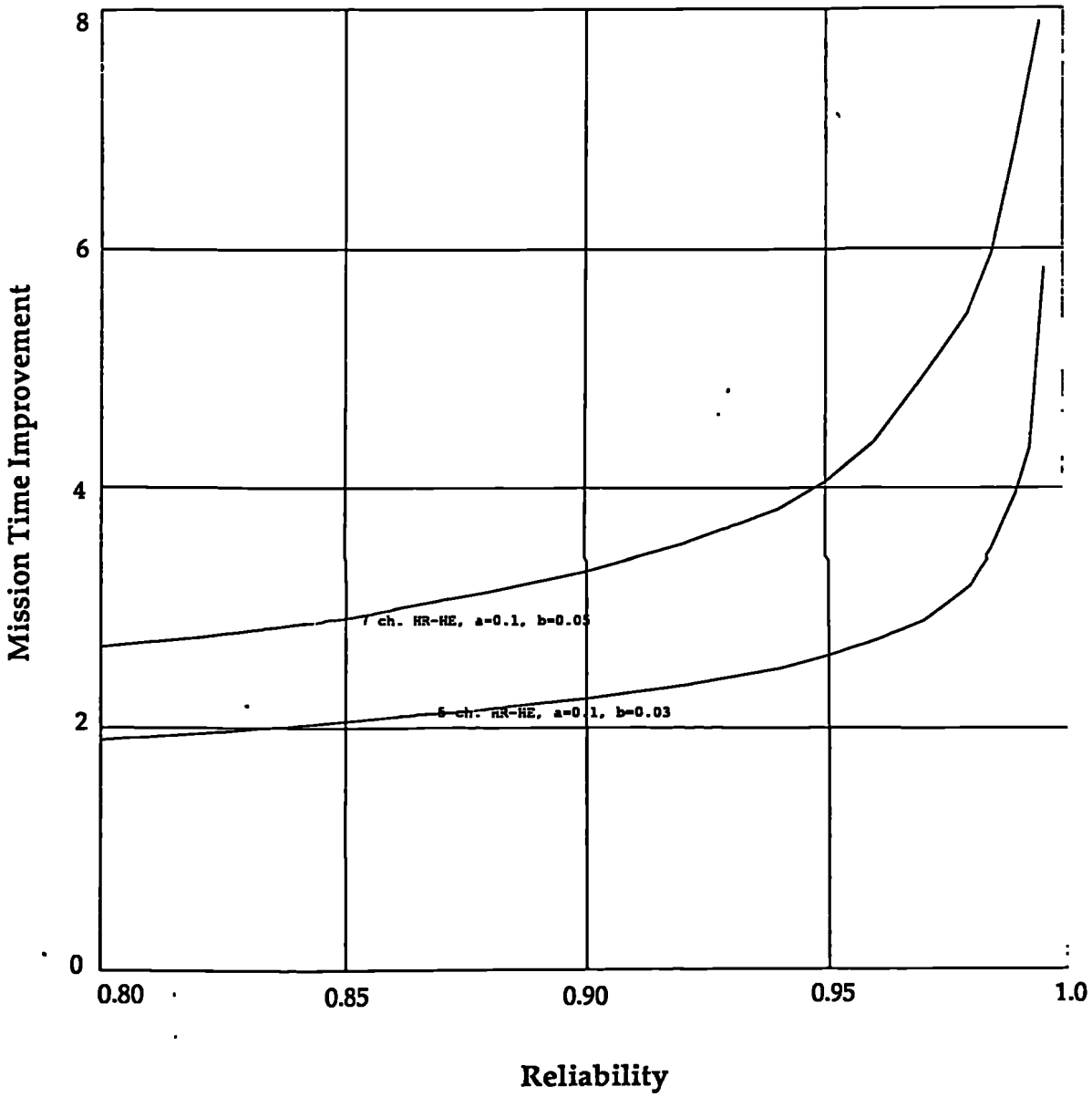


Fig. 5.12 shows the Mission Time Improvement (MTI) of a 5 and a 7 channel HR-HE structures over the TMR structure for different values of α and β.

The failure rate of the basic module is $\lambda = 6 * 10^{-5}$ failures per hours



The MFT-RS scheme compared with The TMR structure:

The overall system reliability of the MFT-RS under the new model with the same assumption for the switch and the voter as in the HR-HE scheme, is given by

$$R_{sys} = \{1 - [\sum_{i=0}^{M} C_i^M (1 - R_m R_m^{\alpha})^{(N-i)} \cdot (R_m R_m^{\alpha})^i]\} \cdot R_m^{c.\beta} \qquad \text{where } M = \frac{(N-1)}{2} - 1$$

An analysis similar to that for the HR-HE was carried out for the MFT-RS . Fig. 5.13 - 5.15 show the graphs of R_{MFT-RS} , and the MTI of the MFT-RS over TMR respectively.

Fig. 5.16 and 5.17 show comparisons between the HR-HE, the MFT-RS schemes, and a hybrid(3,2) which is used an iterative switch array .

Fig. 5.13 System reliability of (3, 5, 7 channels) MFT-RS as a function of time, with perfect voter , switch, and disagreement detection circuit. Time period (t=0 to t=20000 hours).

The failure rate of the basic module is $\lambda = 6 * 10^{-5}$ failures per hours

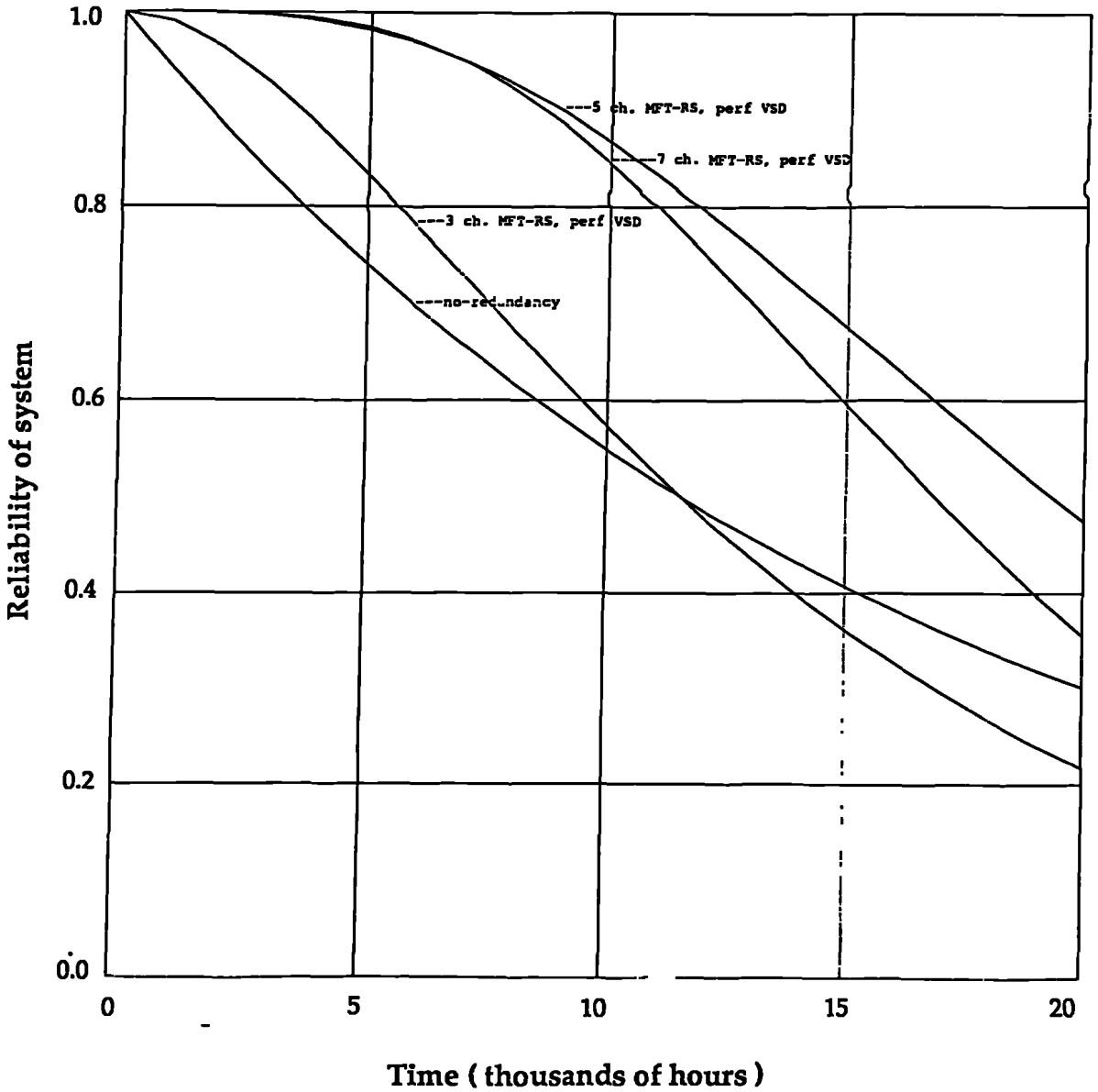


Fig. 5.14 System reliability of (3, 5, 7 channels) MFT-RS as a function of time, with imperfect voter (b=0.02, 0.03, and 0.05 respectively), switch and disagreement detection circuit (a=0.1) as shown in the graph.

Time period (t=0 to t=20000 hours).

The failure rate of the basic module is $\lambda = 6 * 10^{-5}$ failures per hours

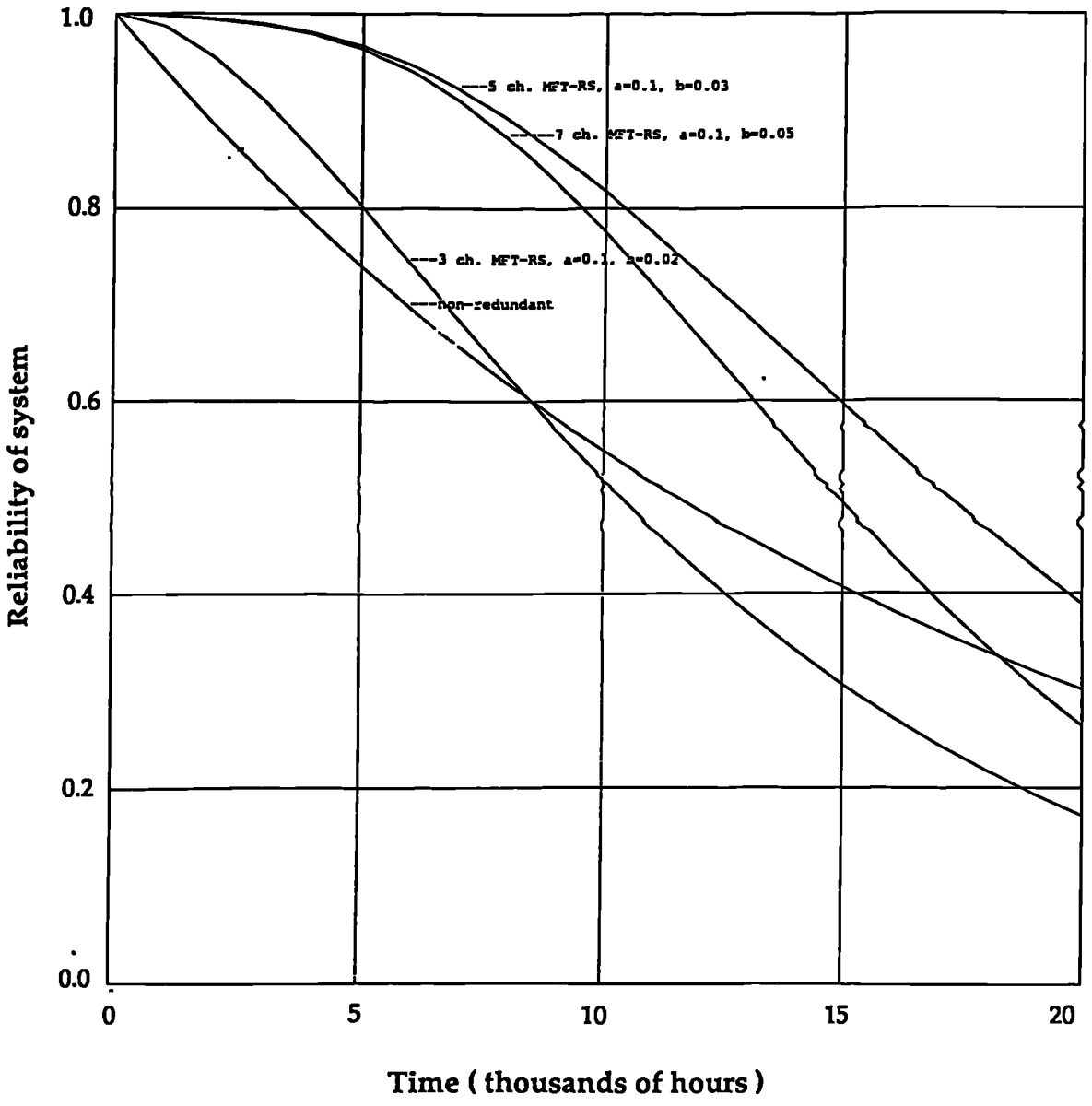


Fig. 5.15 shows the Mission Time Improvement (MTI) of a 5 and a 7 channel MFT-RS structures over the TMR structure for different values of α and β. The graphs show that TMR offers a better mission time for reliabilities close to 1.

The failure rate of the basic module is $\lambda = 6 * 10^{-5}$ failures per hours

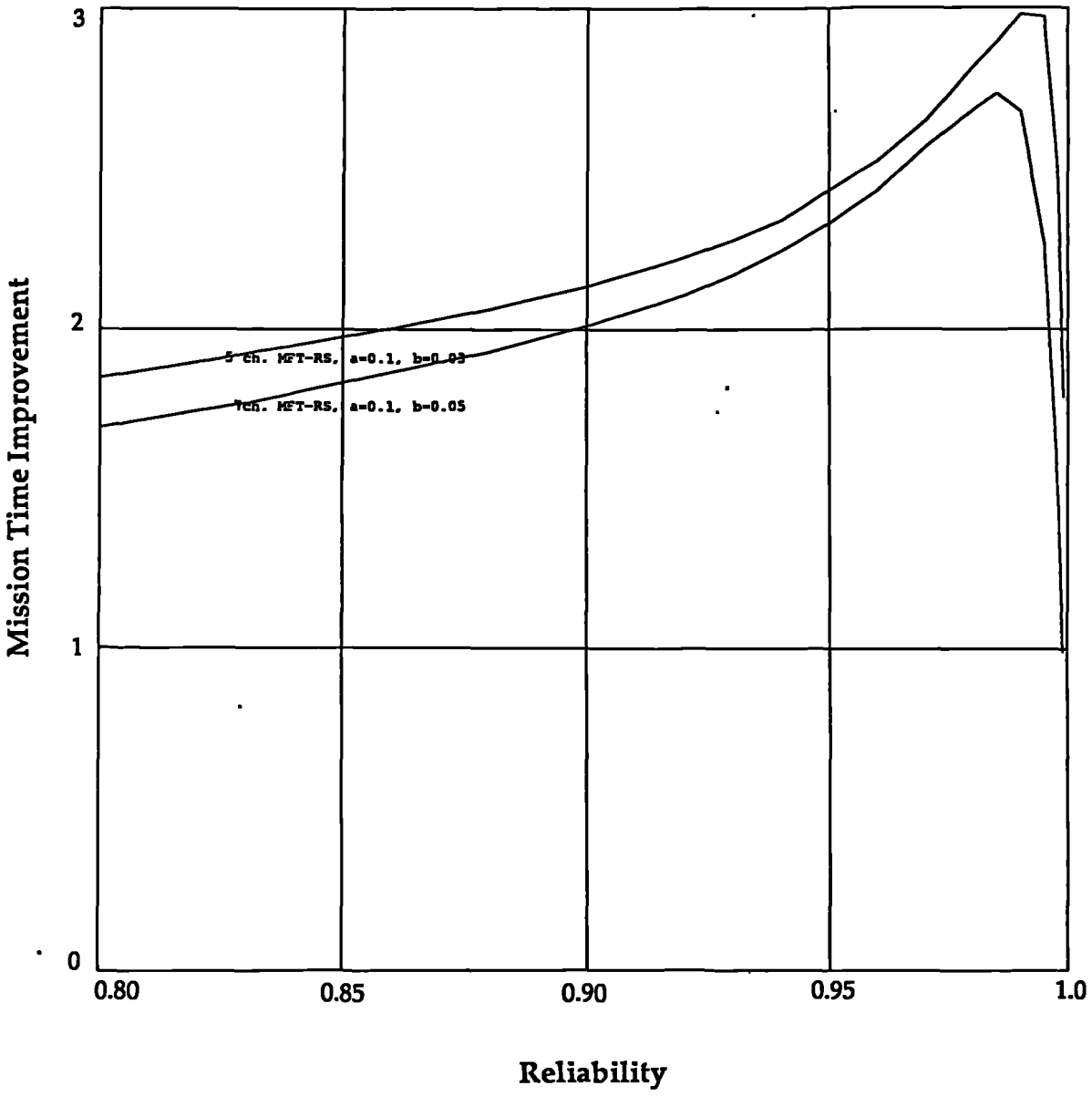


Fig. 5.16 shows the comparisons between the HR-HE, MFT-RS structures, and a hybrid(3,2) (all with perfect switching mechanisms).

The failure rate of the basic module is $\lambda = 6 * 10^{-5}$ failures per hours

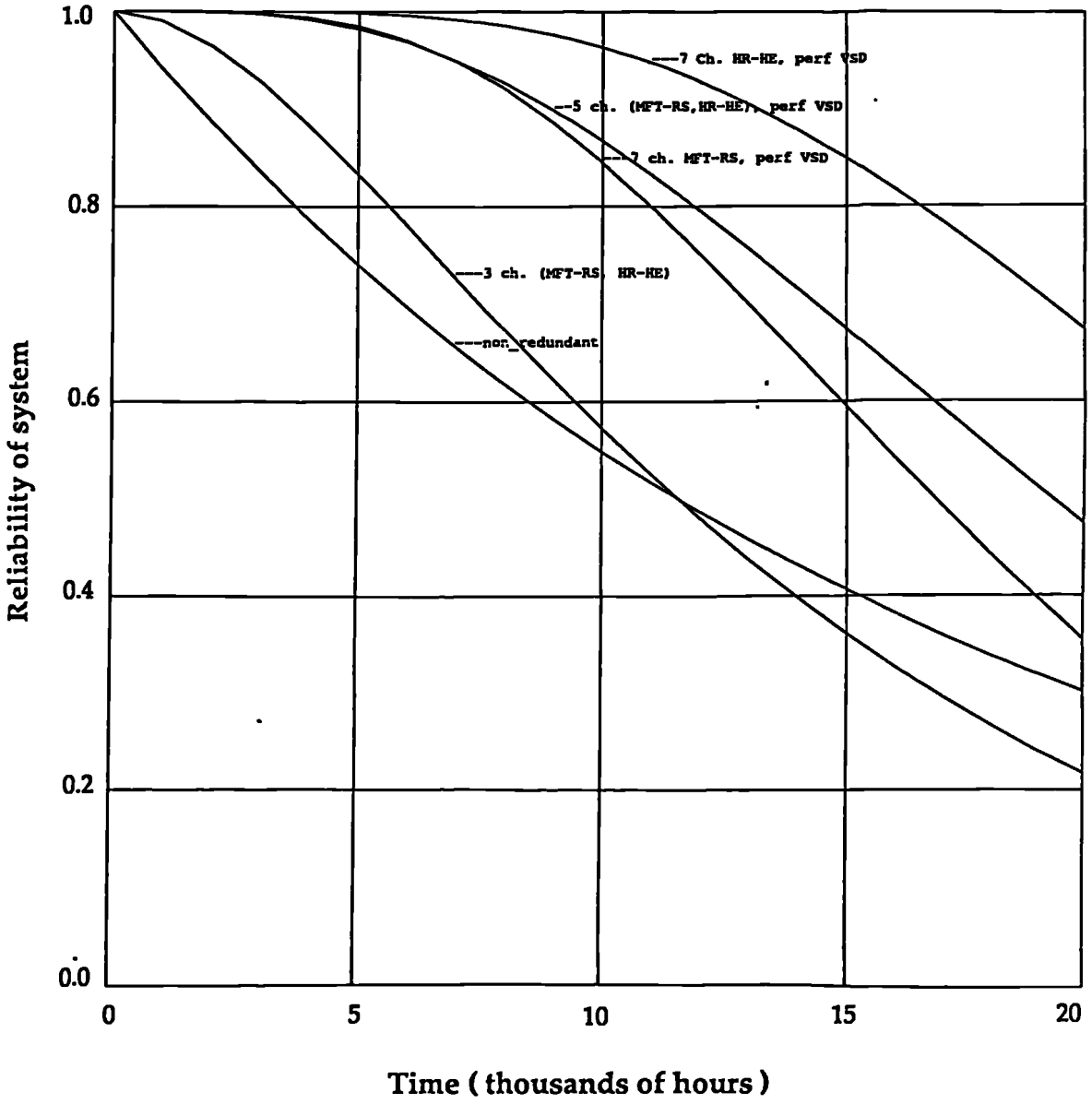
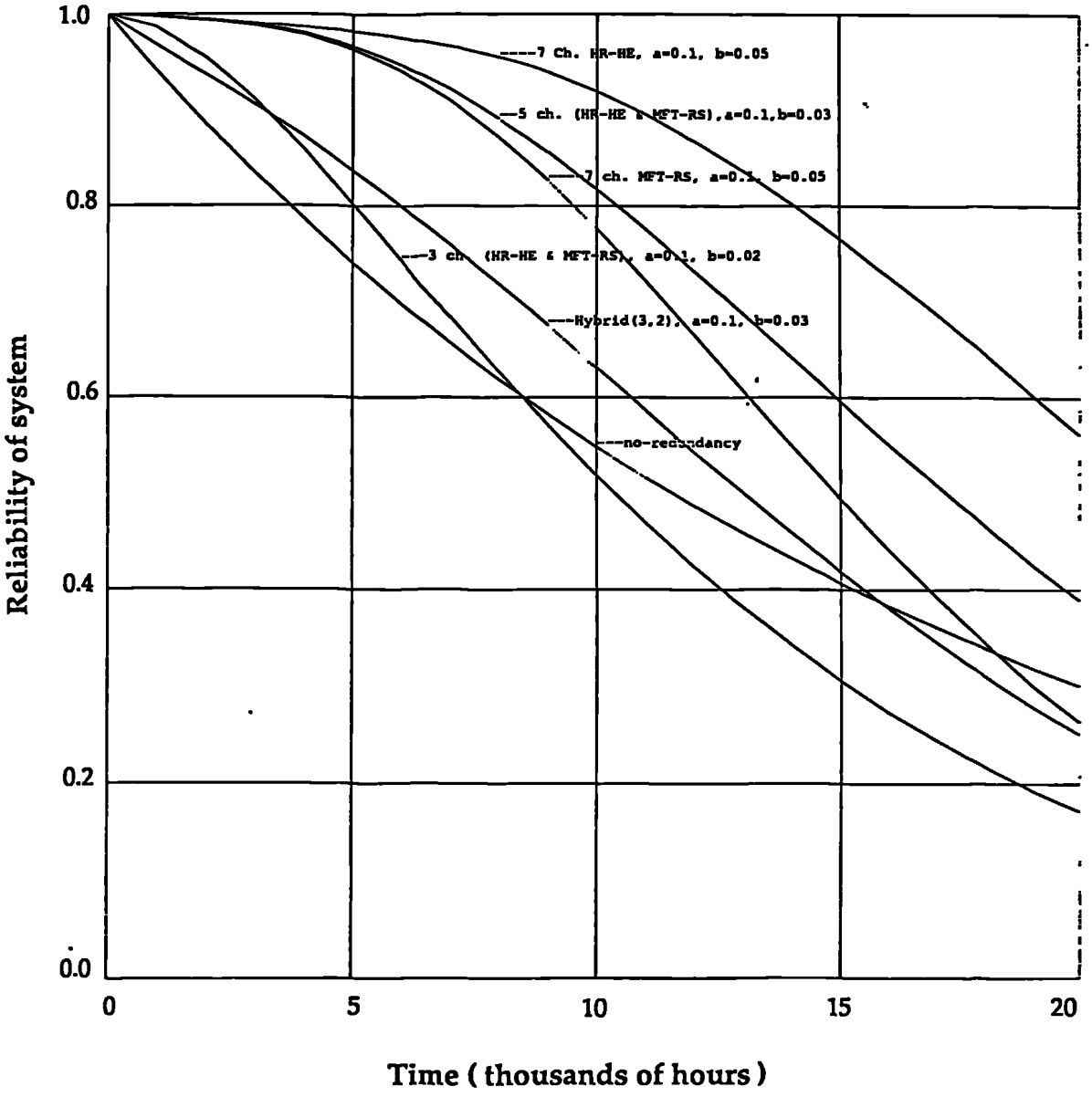


Fig. 5.17 shows the comparisons between the HR-HE, MFT-RS structures, and a hybrid(3,2) with imperfect switching mechanisms and different values for α and β.

The failure rate of the basic module is $\lambda = 6 * 10^{-5}$ failures per hours



5.11 - Critical Components in Fault-Tolerant Structures

Any fault-tolerant system will be totally dependent upon the correct functioning of some critical elements which are usually referred to as the hardcore of the system. In particular, it is vital that the measures and mechanisms provided for fault tolerance are themselves reliable, otherwise little confidence can be placed in the ability of the system to handle failures. For instance, the voter in any of the fault-tolerant designs is regarded as hardcore. In view of its critical role in the system, the hardcore must be designed to operate very reliably indeed. This can only be achieved by adapting fault prevention techniques to minimise the number of faults in the hardcore, or by incorporating further fault-tolerance for the voter itself (as discussed in section 5.4 Fig. 5.3). There are two obvious difficulties . Firstly, at some level the system must be built from components which are not fault tolerant. Secondly, why should it be possible to achieve high reliability for the hardcore of a system any more easily than for the system itself? The solution to these difficulties is simple.

Indeed, simplicity is the key. The hardcore of a well-designed system must be simple; at least it must be much less complex than the rest of the system which depends on it.

In the next section therefore a method of implementing the critical element of the fault-tolerant designs (in our case the voters) will be presented. Two types of voters are generally used ,namely threshold

voters and majority voters. It seems that the majority voters can be realised with less complexity than the threshold voters, particularly when the number of inputs increases. The proposed implementation is for majority voters.

5.12 - Implementation of the majority gates in fault-tolerant designs

One method of implementing the majority gates is , by using the basic gates such as AND/ OR, or NAND gates . Of course using only one type of these gates for the required function of the voters is often very helpful, since a more regular structure can be obtained. But even by taking this approach , too many transistors will be used compared with the following proposed approach, particularly when the number of inputs to the voter are more than three. An improvement in reliability would be expected from a reduction in the number of transistors and the silicon area utilised for the circuits.

In order to conserve physical space on the chip it is most important to employ *regular structure* in designing circuits and in particular majority voters. Implementing the voters using basic gates , e.g. NAND gates, (in its equivalent MOS transistor circuit form) would be unstructured and hence inefficient in terms of silicon area utilisation and the number of interconnections required. What is required is a cellular or modular structure which can be used by itself as a voter or be repeated (cascaded) to achieve the required function of the voting.

5.13 - Modular approach for the voter implementation

The developments in integrated circuit technology , particularly the use of MOS transfer gates (also called a pass transistor) has made an old technique called iterative network [Mu86], attractive again. Application of the pass transistor and the use of the above technique is the basis for implementation of highly reliable voters.

The n-MOS pass transistor acts as a switch when there is a positive voltage on its gate . The switch is closed or 'ON' if the drain and source are connected. This can be achieved by applying a voltage to the gate (gate HIGH). The switch is open or 'OFF' if the drain and source are disconnected which can be achieved by zero voltage to the gate (gate LOW) . The schematic representation is shown in Fig. 5.16

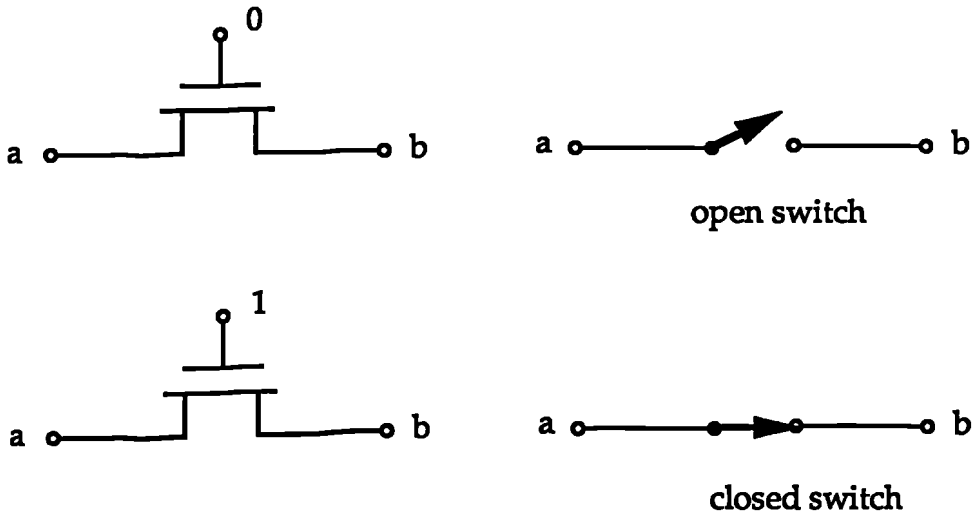


Fig. 5.16 shows a MOS transistor as a switch.

There is however a disadvantage of using pass transistors and that is the delay encountered when they are connected in series . In this case a degradation of the signal level will occur which should be reinstated by the use of invertor amplifier . If n pass transistors are used in series , the delay is proportional to n^2 , and for n-MOS transistors the maximum number of transistors in series is about four , but using CMOS technology will increase this number. However the delay that occurs for small circuits such as the voters in fault-tolerant schemes is less than the delay that occurs when basic gates are used.

There is another point about the function of the voters which helps to simplify their implementation. It should be noted that the majority function is a symmetric function. A symmetric function is a logic function which has the property of remaining unchanged when any two of its variables (called the variables of symmetry) are interchanged. For example , the following function is symmetric, since

$Z = X_1 X_2 + X_1 X_3 + X_2 X_3$

if the variables X_1 and X_2 are interchanged (i.e. replace all X_1's with X_2's and vice versa) we obtain

$Z = X_2 X_1 + X_2 X_3 + X_1 X_3$

which is identical to the original function .

Symmetric functions can be represented by a basic contact (path closing) network, which has one input and which branches out to give m+1

outputs, where m is the number of variables . Symmetric circuits are of considerable importance in LSI/VLSI design , since the contact network may be mapped directly into NMOS circuitry .

The circuit is such that a logic 1 (HIGH) signal will propagate through the network from the pull-up transistor to an output, with the particular path being defined by the states of the input variables (a high signal effectively closes the path) ; logic 0 signals will propagate from ground to all other outputs . Additional pass transistors are needed to obtain the required function . Fig 5.17 illustrates the implementation of a three inputs majority gate . The circuit operates as follows:

if all the input variables (X_1, X_2, X_3) are HIGH then the transistors on columns 2, 4, and 6 are ON and pass the high voltage (applied to the depletion mode pull-up transistor) to the top-most output line in the diagram. So, any time that there is a high voltage on this line , all of the three input variables are HIGH (logic value 1) . Similarly, if there is a HIGH voltage on any of the other output lines, the number under the output line represents the number of HIGH input variables. Now the output of the voter is assumed to be 0 unless a HIGH voltage is on the output line 3, or 2, which indicates that the voter output should be logic 1.

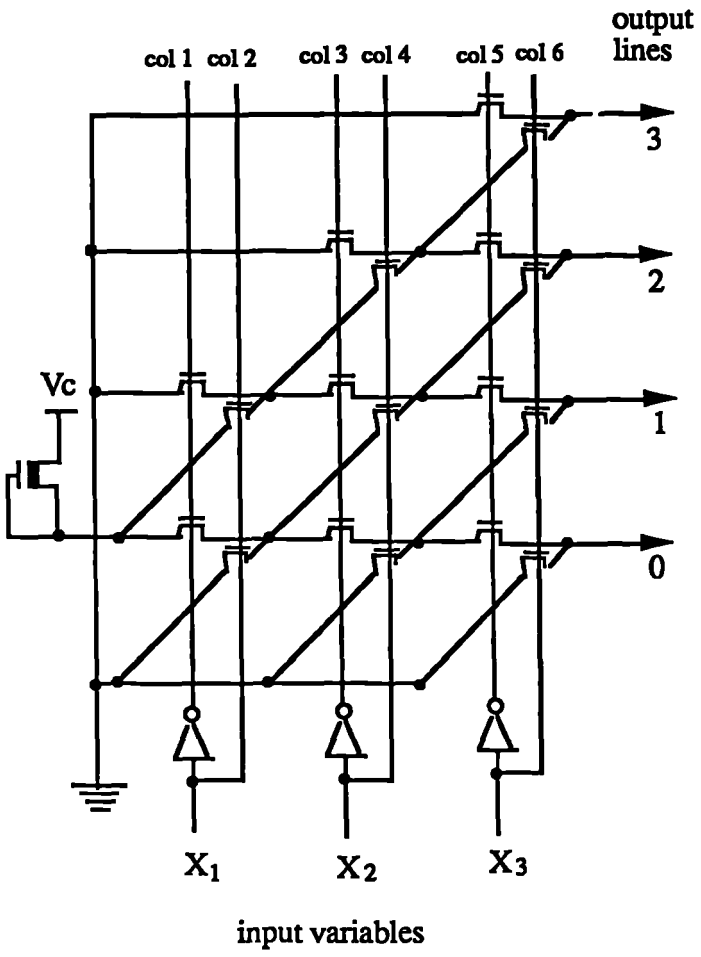


Fig. 5.17 The implementation of a 3 input majority voter using n-MOS transistors

It is possible to optimise the number of transistors in this design. As we are not interested in the value of outputs 1, and 0, the transistors passing the voltage to these lines are not needed and may be omitted. This is shown with the dotted box in the schematic diagram of the above design in Fig. 5.18.

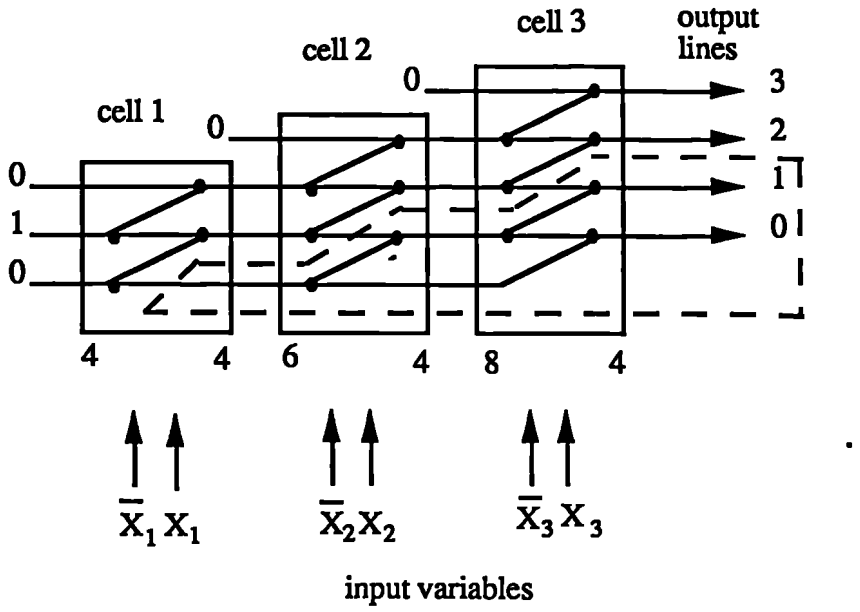


Fig. 5.18 The schematic diagram a 3 input majority voter (modular approch)

In the above diagram each sloping or horizontal line within each cell represents one pass transistor, with the exception of the bottom-most horizontal line which is a connection to the ground. The number of transistors before the optimisation are shown on the bottom-left corner of each cell , and after the optimisation , on the bottom right corner.

Fig. 5.19 and Fig 5.20 show a five inputs , and a seven inputs majority gate respectively.

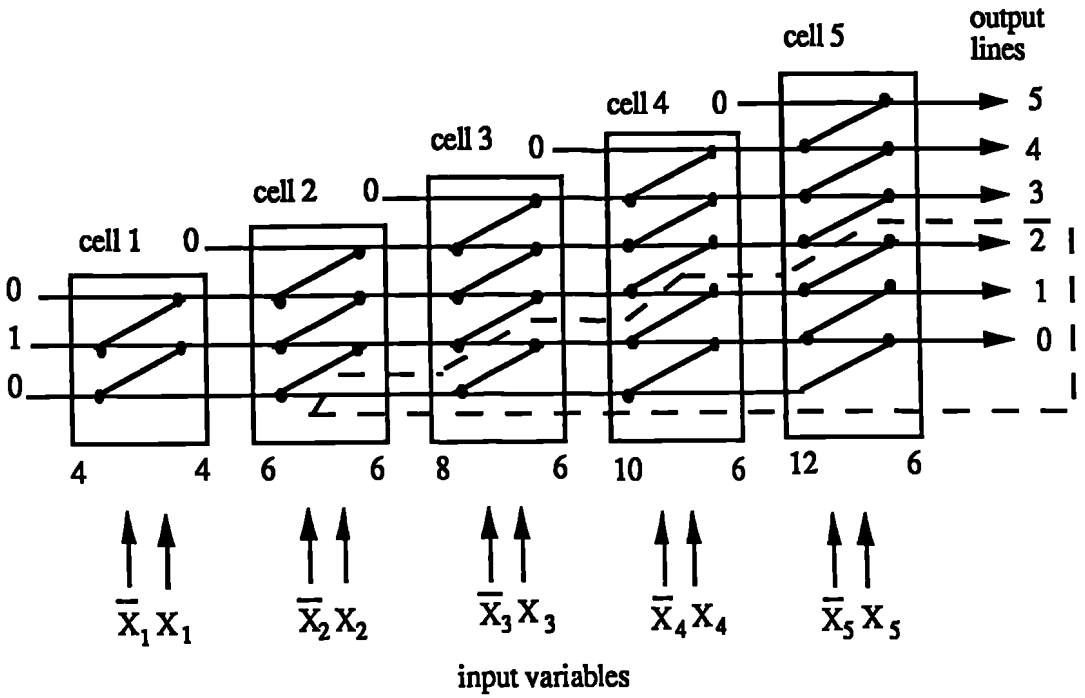


Fig. 5.19 The schematic diagram for a five input majority gate

The reliability of each voter also can be calculated in terms of the reliability of each transistor. If the reliability of each transistor is denoted by R_{tr} , the reliability of the voter may be expressed by the following equation

$R_v = (R_{tr})^n$ where n is the number of transistors used in the voter.

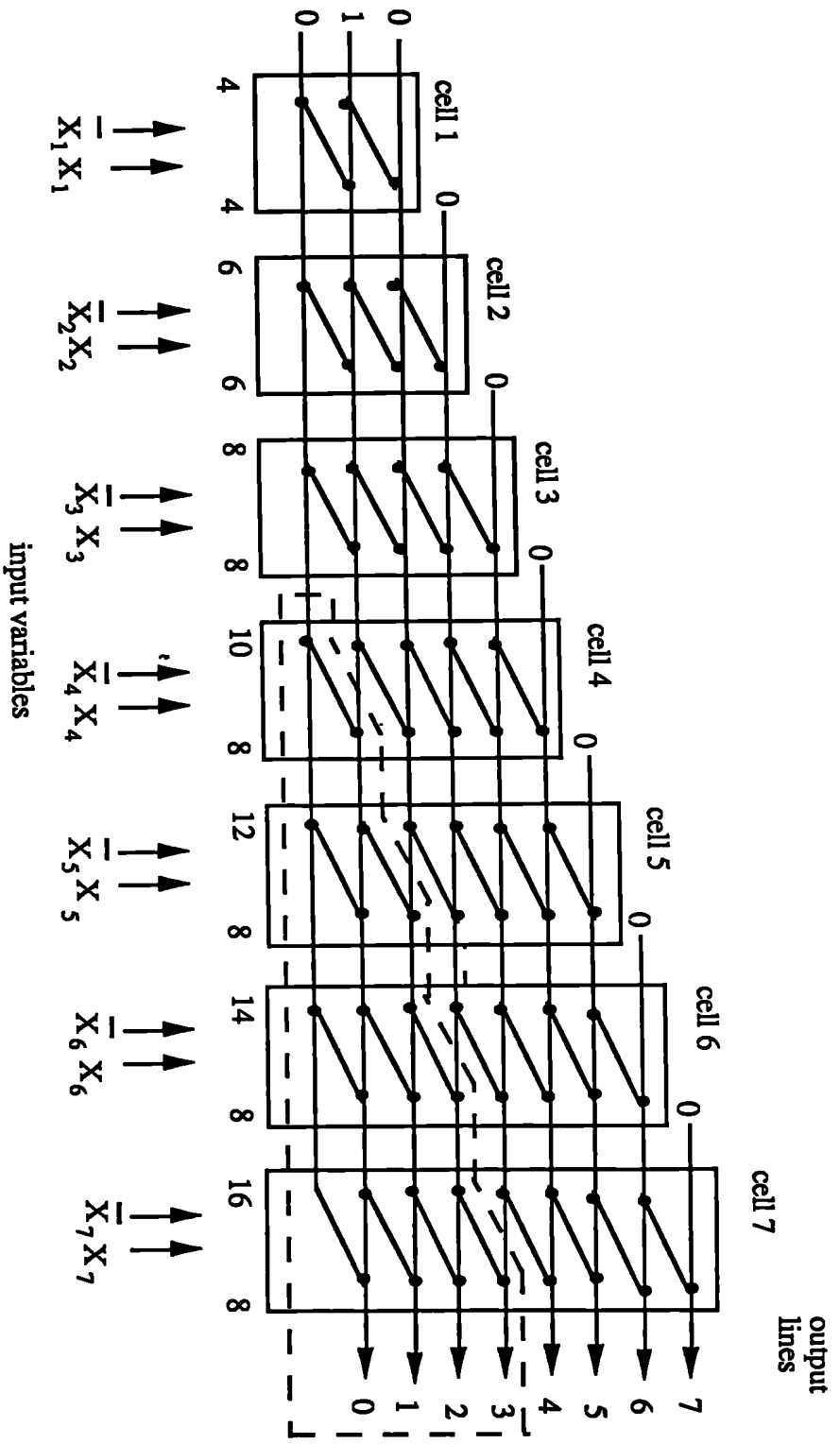


Fig. 5.20 The schematic diagram of a seven input majority voter

The following table shows a comparison of the number of transistors used for each voter using the above approaches .

Majority voter no. of transistors	3 Inputs gate	5 Inputs gate	7 Inputs gate
Implemented by NAND/NOR	16	86	374
Implemented by NMOS pass transistors *	25	51	85
Implemented by NMOS pass transistors **	19	39	65

* Transistors ' number befor optimization

* * Transistors ' number after optimization

Table shows the number of transistors used for the implementation of 3, 5, and 7 inputs majority gates by different approaches.

5.14 - Summary

In this chapter the overall reliability of a few fault-tolerant designs (including our new designs) has been analysed to show when and how much reliability improvement can be achieved by the different techniques. To do these analyses, the classical reliability model was first discussed and used. The case of the TMR structure was discussed in detail because TMR is often used as a threshold to compare different redundancy techniques. Secondly a new reliability model was developed . It was shown that because of the structure of the switches

in the MFT-RS and the HR-HE schemes, each module and its related switch component can be modelled as a pair. Therefore the new reliability model can be used for these schemes. The results of these analyses were plotted and it was shown that the new fault-tolerant techniques have a few advantages over the other well known techniques.

Finally in this chapter a new approach was proposed to reducing the size and the complexity of voters in fault-tolerant designs. Since voters are the most critical components in fault-tolerant systems, decreasing their complexity has a benficial effect on overall system reliability as well as other factors such as area overhead and power consumption. It was shown that a great reduction in the number of transistors can be achieved by the application of n-MOS pass transistors within a modular structure for a voter (particularly when the number of inputs to the voter is greater than three).

CHAPTER SIX

APPLICATION OF EXPERT SYSTEMS IN FAULT-TOLERANT DESIGNS

6.1 - Expert systems

6.1.1 - Introduction

In this chapter, expert systems will be discussed from the point of view of an application to a fault-tolerant design choice assistant and prediction of reliability of digital systems. Expert systems are used in an attempt to minimise or eliminate the needs for highly specialised experts in this field. Experts are people who are very good at specific types of problems. Their skill usually comes from extensive experience, and detailed specialised knowledge of the problems they handle, for example engineering experts who carry out diagnosis and repair of high technology equipment, such as computers.

Whenever human experts posses complex knowledge about a highly specific subject area and are in great demand and short supply, a computer-based consultant can help, amplify, and disseminate the needed expertise. A computer-based expert system seeks to capture enough of the human specialist's knowledge so that it too will solve problems expertly, freeing the human experts to concentrate on areas that are more useful.

One popular application area for expert systems is microelectronics. In sophisticated VLSI circuit design, such as fault-tolerant designs, there is likewise a shortage of trained experts, and as technology becomes more complex, this problem will get progressively worse. There is also a shortage of expert reliability engineers to evaluate and predict the reliability of these complex circuits under different conditions, at

different times, and with a wide range of variables.

Expert systems that perform prediction infer the likely consequences of given situations. Prediction systems sometimes use simulation models, programs that mirror real-world activity, to generate situations or scenarios that could occur from particular input data. These potential situations, together with knowledge about the processes that originated them, form the basis for the predictions. It should be pointed out that relatively few prediction systems have been developed to date, possibly because of the difficulty in creating and interfacing with simulation models. Prediction of system reliability is one area which require more attention. Therefore a computer-based system is needed and can be developed to predict digital systems reliability.

The new tool (expert systems in this field) has its greatest value in that it predicts the overall reliability of systems under different conditions at any required time.

6.1.2 - Why build an expert system?

We have mentioned the most obvious reasons: dissemination of rare and costly expertise, and the more effective and efficient use of the human expert.

Other reasons for building expert systems are:

- the possibility of combining the expertise from many human experts into a shared knowledge-base that can be then studied for consistency and reliability of its advice.
- the permanence of these systems. Human expertise can quickly fade,

regardless of whether it involves mental or physical activity. An expert must constantly practice and rehearse to maintain proficiency in some problem area. Any significant period of disuse can seriously affect the expert's performance.

- the ease with which an expert system can be transferred or reproduced. Transferring knowledge from one human to another is the laborious, lengthy, and expensive process called education or training. Transferring artificial expertise is the trivial process of copying or cloning a program or data file.

Thus expert systems are suitable for tasks that require experience in order to perform them proficiently. Gaining experience is a time consuming, and often a very expensive process.

Thus unlike a human expert whose time is restricted, an expert system is available for use 24 hours a day, every day of the year. Human experts are limited in number, while many expert systems can be created. In addition the computerised expert system never dies - taking the knowledge with it -, the knowledge in an expert system can be easily copied and stored , thus loss of knowledge is actually quite rare. Finally expert systems are always at peak performance, and often compute the best possible opinions (within the limitation of their knowledge), while a human expert gets tired, affecting the reliability of his advice in some cases. For the above reasons an expert system will be implemented. In this chapter we describe how an expert system may be designed and used as a fault-tolerant design adviser.

6.2 - Development of an expert system

Generally there are three groups involved in the development of an expert system

I) Experts

This group provides the specialised knowledge for the expert system.

II) Knowledge engineers

This group questions the experts, structures the knowledge and uses it to implement the knowledge base.

III) Users

They state their requirements and ideas, and above all, define the scenario in which the expert system will be used.

In the development phase, the main emphasis is on the work of the knowledge engineer and the expert [Wa86]. Their relationship is shown in Fig 6.1.

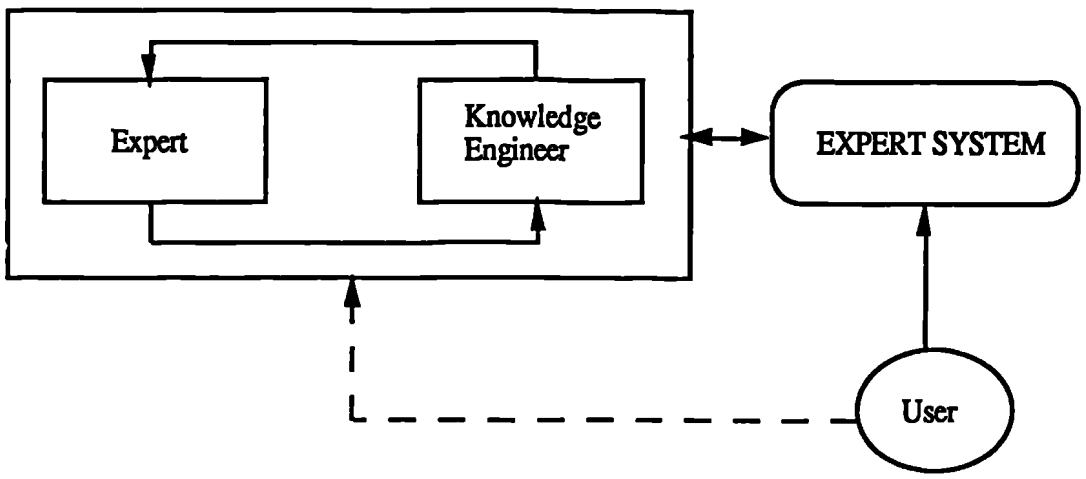


Fig. 6.1 The relationships in the Development of an Expert System.

6.3 - How expert systems operate

Expert systems are computer programmes that can mimic the behaviour of a human expert. An expert system will use information supplied by the user to give an opinion on a certain subject, or it will ask questions until it can identify an object that matches the answers. Every expert system has two major parts : the knowledge base and the inference engine [HaWaLe83] [We84].

6.3.1 - The knowledge Base

The knowledge base is a database that holds specific information and rules about a certain subject. There are two basic terms that will be used frequently in our discussion:

OBJECT: The conclusion that is defined by its associated rules

ATTRIBUTE: A specific quality that, with its rule, helps define the object .

Therefore, our knowledge base contains a list of objects with their associated rules and attributes. In the simple sense (and for many applications), the rule that is applied to an attribute states that the object either "has" or "has not" the attribute. Thus, an object can be defined by using a list of attributes that the object either does or does not possess. For example, an expert system that may advise on various types of redundancy techniques might have a knowledge base like this:

OBJECT	RULE	ATTRIBUTE
TMR	has	3 basic modules
TMR	has	a 3 input voter
TMR	has no	switching circuit

The knowledge base can be simplified more by using only one rule (e.g. "has"), and a negative form of the attribute will be used if a "has no" relationship should be established. Thus, the rule simply becomes "possesses," and the simplified knowledge base looks like this:

OBJECT	POSSESS
TMR	has 3 basic modules
TMR	has a 3 inputs voter
TMR	has no switching circuit

Although sophisticated expert systems may need more complex rules than simply "possesses," this rule is sufficient for our situations and greatly simplifies the knowledge base. Our system is implemented such that the knowledge base consists of only objects and their attributes.

6.3.2. - The inference Engine

The inference engine is the part of the expert system that attempts to use the information that the user supplies to find an object that matches. There are two broad categories of inference engines:

deterministic and *probabilistic* [Na87].

In the deterministic category, the user will get his answer with certainty, (as for example, the number of electrons in an atom which express the identity of that atom). However in the probabilistic category the answer has a degree of uncertainty. That is the answer is likely, but uncertain, as for instance, the position of the electron around an atom, when its velocity is already calculated (uncertainty principle). Most disciplines are not deterministic, but rather are probabilistic to a certain extent. However, for many of these, the uncertainty factor is not statistically important so they can be treated as deterministic situations. In our case we deal only with a deterministic expert system because its implementation is easier. Beyond the two broad categories of certainty and uncertainty, there are three basic ways to construct the inference engine: forward chaining, backward chaining, and rule value [Ne87]. The differences of these methods relate to the way that the engine attempts to reach its goal.

6.3.3. - The Forward-Chaining Method

Forward-chaining is sometimes called data-driven because the inference engine uses information that the user provides to move through a network of logical ANDs and ORs until it reaches a terminal point, which is the object. If the inference engine cannot find an object by using the existing information, then it requests more. The attributes that define the object create the path that leads to the object (the only

way to reach the object is to satisfy all of its rules) . Thus, a forward-chaining inference engine starts with some information and then tries to find an object that fits the information.

Fig. 6.2 shows the structure of the forward chaining. A forward-chaining system essentially builds a tree from the leaves down to the root.

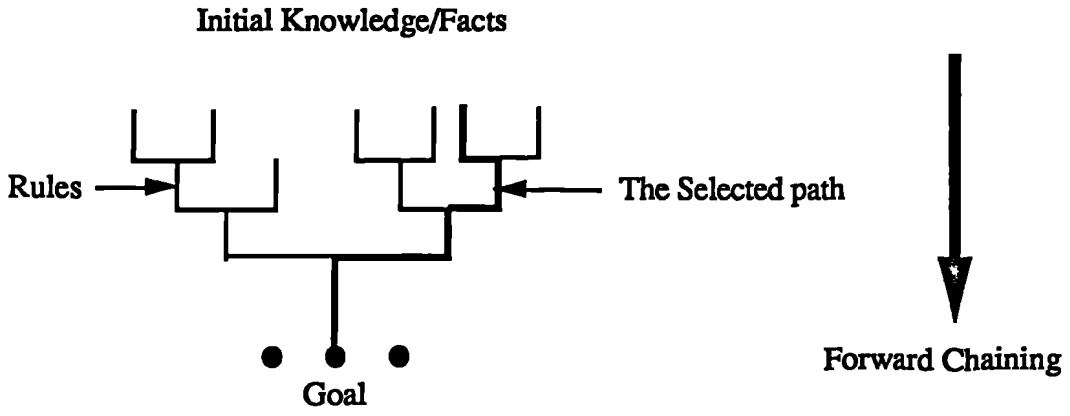


Fig. 6.2 Forward-chaining , shows a path through the decision tree (from the attributes to the object).

6.3.4. - The Backward-Chaining Method

Backward-chaining is the reverse of forward-chaining. A backward chaining inference engine starts with a hypothesis (an object) and requests information to confirm or deny it. Backward-chaining is sometimes called object-driven because the expert system begins with

an object and attempts to verify it. To show how the backward-chaining works, imagine that our computer suddenly stops working. The first hypothesis for instance is that it has lost power. To check this, we listen for the fan. Hearing the fan run, this hypothesis will be rejected and we will proceed to another. The second hypothesis is that our computer has crashed because of faulty software. To confirm or reject this possibility, the computer will be rebooted and it will work, so the second hypothesis holds true, and the answer will be obtained.

Fig. 6.3 illustrates the backward chaining method . As the diagram shows, backward-chaining prunes a tree. This is the process opposite to that of forward-chaining, which construct a tree.

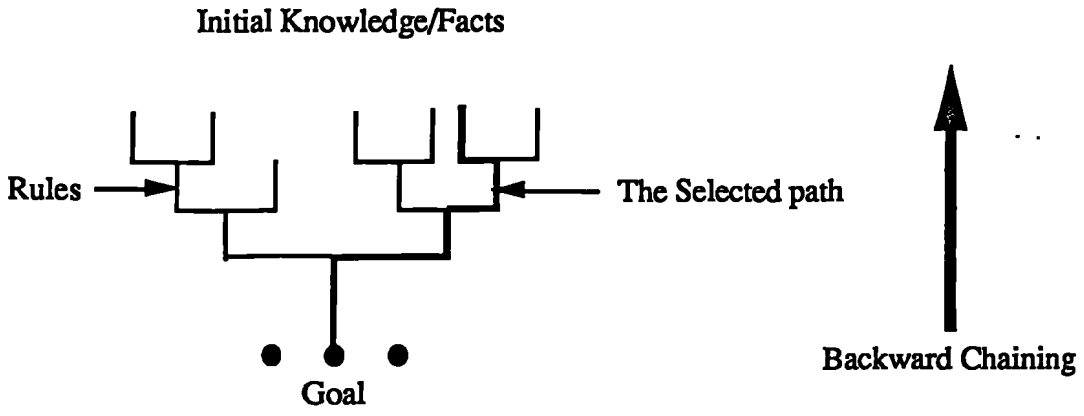


Fig. 6.3 backward-chaining , shows a path through the decision tree (from hypothesis object to the attributes).

6.3.5. - The Rule- Value Method

A rule-value inference engine is theoretically superior to either a forward-chaining or backward-chaining system because it requests information that has the greatest importance according to the current state of the system. A rule-value inference engine is actually an improved backward-chaining engine. The general operating theory is that the system requests as its next piece of information the one that will *remove* the most uncertainty from the system. In this approach, the key point is the selection of those questions that make the most rapid progress to a conclusion. The trouble with rule-value systems is that they are difficult to implement. There are two reasons for this:
First : in real-life situations, the knowledge base is often so large that the number of possible combinations exceeds what a system can easily hold. Hence, the system cannot know what information removes the most uncertainty for any given state. *Second* : rule-value systems require the knowledge base to contain not only the standard object-attribute information, but also a value quantifier, which makes constructing the knowledge base more difficult. However, there are certain situations that lend themselves to rule-value inferences more than others. Also, when implemented, rule-value systems generally do a better job than the other two methods. Note that some rule-value expert systems began as either forward chaining or backward-chaining systems that had a statistical module to record various aspects of the system. Later, after an expert system of that type has been used awhile,

this statistical information can be used to implement a rule-value approach.

6.4 - Choosing a Method

At this point, the question is, which of the three types of inference engines is the best to be used? The answer is, all three types can do the job. But as stated earlier, a rule-value system is more difficult to be implemented, so as we like to built a simple expert system, at this stage, we should probably avoid this method. The forward-chaining method makes the process of deriving the greatest amount of information from the knowledge base somewhat easier because it constructs a tree. A typical forward-chaining system finds all possible objects that match the attributes. The advantage of the backward-chaining method is that it requests only enough information to find an object. Thus, because backward-chaining systems are goal-driven, they allow only relevant information to be input into the system. A backward chaining system is good when we want only one object- even if other objects also satisfy the attributes. It is possible to create a backward-chaining expert system that finds multiple solutions, but it does require more work than constructing a forward-chaining expert system. In the final analysis, any of the above approaches may be used. However in this chapter, the backward-chaining method will be used and an expert system will be implemented which can be developed later by using rule-valued method .

6.5 - Creating the Expert System

Now that we have the necessary background on expert systems, an expert system that uses backward-chaining will be created in this section. The implementation includes the necessary routines to create the knowledge-base. The structure of our expert system is such that the knowledge-base is separated from the inference engine (this helps us avoiding some serious troubles which may occur by mixing them) .

To create the inference engine it is assumed that the knowledge base consists only of objects and their attributes. The following specifications also have been considered in the creation of an efficient inference engine:

- The expert system must not inquire about the same attribute twice.
- The expert system should reject immediately and move past any object that does not have one or more of the known necessary attributes or that has an attribute that has already rejected.
- Upon command , the expert system should be able to report why it is following a line of reasoning.

The third constraint not only is a way of verifying that the expert system is operating correctly , but also is a method of educating the user.

6.6 The structure of the expert system

In this section the expert system will be described in the form of

structure charts.

6.6.1 The main body

At the highest level the system consists of five main subfunctions:

- the Enter ( ) subfunction
- the Load ( ) subfunction
- the Query ( ) subfunction
- the Save ( ) subfunction
- the eXit ( ) subfunction

which can be selected by the letter capitalised in each function. The operation of the functions will be defined later. The structure chart is shown in Fig. 6.4 which shows the hierarchy of the functions within this level.

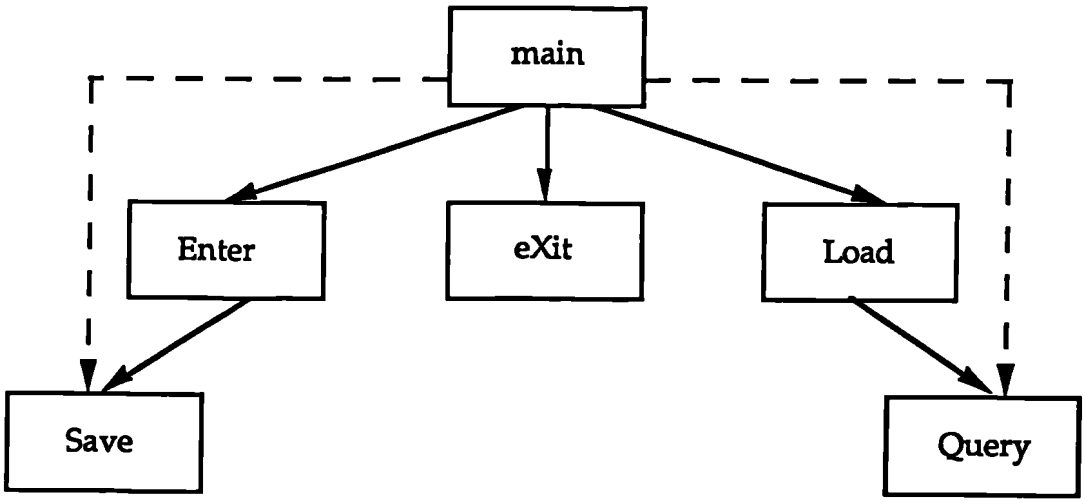
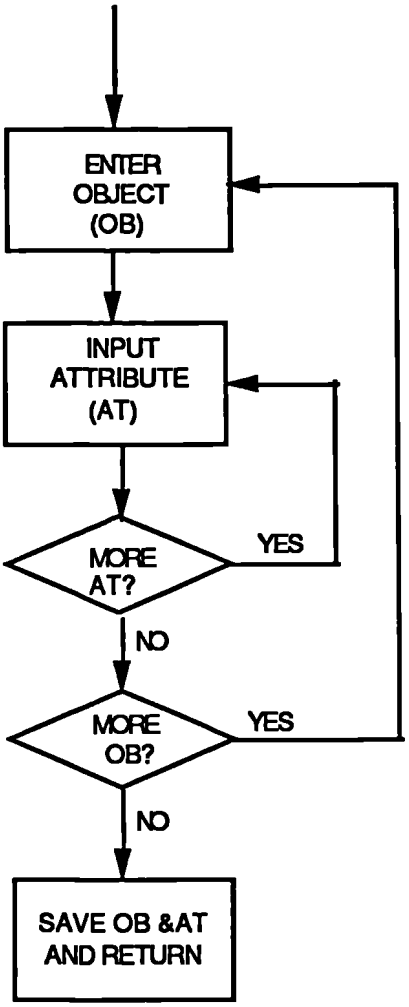


Fig. 6.4 Illustrates the structure of the main

6.6.2 Entering the Knowledge into the Knowledge Base

The *enter* procedure provides a set of facilities which allow the expert to add expertise into the knowledge-base. This information should be in the form of objects' names and their attributes. Fig. 6.5 illustrates the chart for this procedure. The procedure starts by pressing letter E for enter. Then the system will ask the expert to enter the name of the object to be included in the object list of the knowledge base. Then it would ask for the attributes of that object. These attributes will be included to the attribute list. By pressing the return key at the question for the next attribute, (if there is no more attribute for the object), the system will ask for another object. If the return key is pressed again, (when no more object is required to be included), the procedure will be ended and the function returns to the main menu. At this stage the entries to the knowledge base should be saved. Thus function *save* will be called, which saves this information in a data file called *expert.dat* .



ENTERING OBJECTS (KNOWLEDGE)

INTO THE KNOWLEDGE BASE

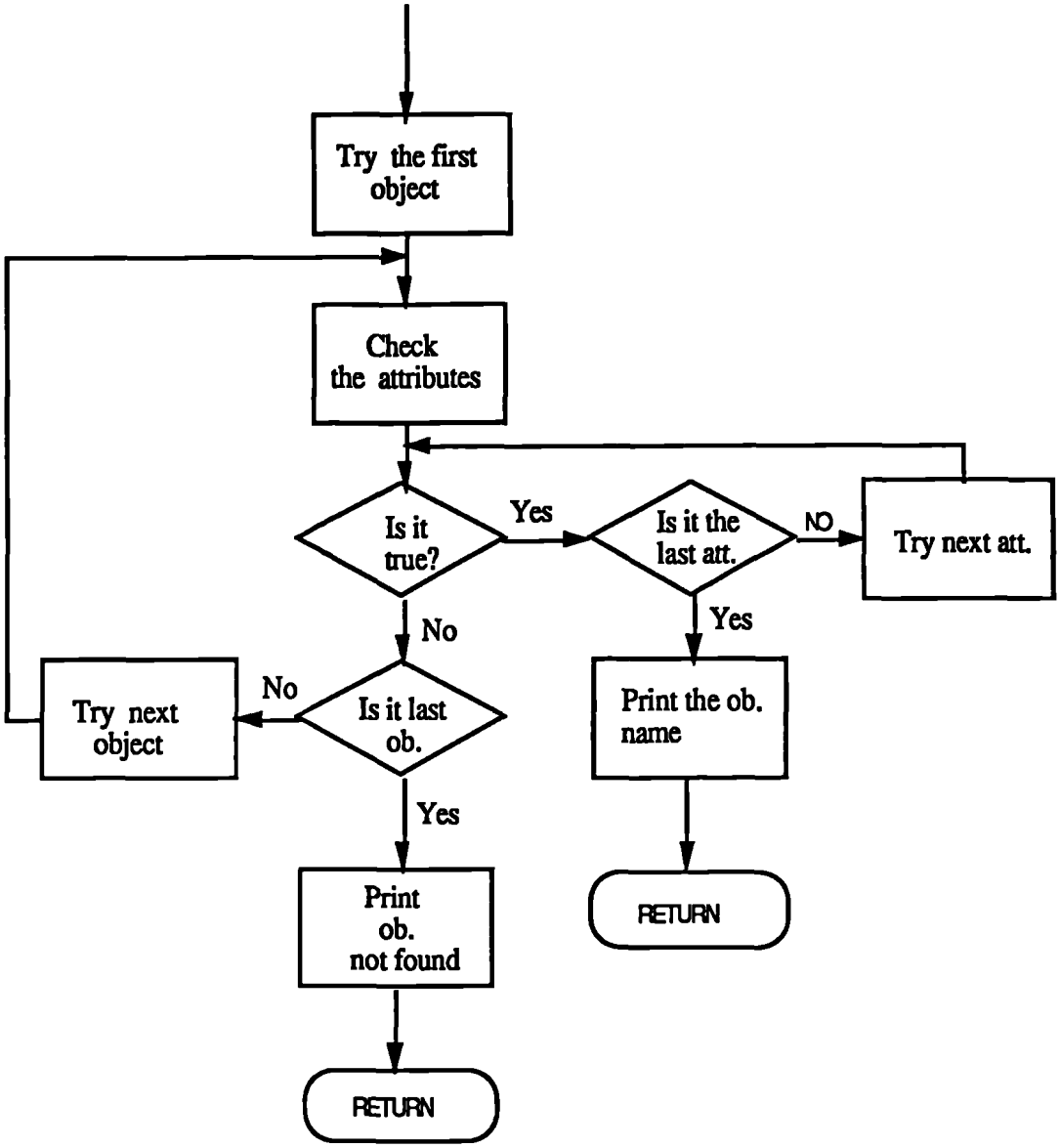
Fig. 6.5 Shows how the Knowledge Base acquires information

6.6.3 Using the Inference Engine

The other part of the structure is the inference engine. The *Query* procedure describes the operation of our inference engine. Chart 6.6 illustrates how the engine queries about an object and its attributes. This function will not begins the query unless the knowledge base is already loaded into the system. The *Load* function provides this facilities when it is called by the main, by pressing letter L. When the knowledge is loaded into the system, the Query procedure functions by pressing letter Q at the main.

As shown in Fig. 6.6 , the engine selects the first object at the top of the object list in the knowledge base, and checks for its attributes. If all the answers to the questions are *true* , then the system prints out that object as the conclusion of the query. If any of the answers is *false* , then the next object in the list will be selected. This procedure will be continued until the object which matches best with the answers can be selected. In the case that the system can not conclude the correct answer , the appropriate message will be printed.

Examples of entering knowledge into the system and loading and query about an object are given in Appendix B.



THE STRUCTURE OF THE INFERENCE ENGINE

Fig. 6.6 Illustrates the process of query about an object

6.7 - Summary

In this chapter we have discussed the development of an expert system to be used as an advisor in fault-tolerant techniques. Generally three groups should be involved in the development of an expert system (experts, knowledge engineers, and users), but in the development phase the main emphasis is on the work of the knowledge engineer and the expert.

An expert system has two major parts: the knowledge-base and the inference engine. The knowledge-base holds specific information and rules about a certain subject (fault-tolerance techniques) in terms of objects and their attributes.

The inference engine attempts to use the information that the user supplies and the knowledge in the knowledge-base to find the answer to the query.

Three general methods for the implementation of inference engine were discussed (the forward chaining, the backward chaining, and the rule-value method) and their differences were shown. The system developed in this research has used the backward chaining method. After gathering enough information the system can be developed and a rule-value method may be used.

The result of the system run is shown in appendix B as an example.

CHAPTER SEVEN

CONCLUSION AND FURTHER WORK

Conclusions

7.1 -

Rapid advances in integrated circuit (IC) technology have enabled the fabrication of digital circuits with a very large number of devices on a single chip. With the help of CAD/CAE techniques, design and fabrication of ICs have advanced to the point that functioning ICs of considerable complexity can be produced in one relatively quick iteration (IEEE). Unfortunately, some circuits on a finished wafer will not work properly, while others will suffer packaging flaws.

On the other hand there is an ever increasing demand for highly reliable computation, highly available systems, and long-life applications. These requirements can be achieved through two general approaches, *Fault-prevention* and *fault-tolerance.*

Fault prevention techniques are employed prior to the construction of a system. There is an upper limit to reliability improvement that can be achieved using these techniques due to technology and cost limitations. Any attempt to improve this limit is either very costly or produces poor results.

Fault-tolerance is the other approach to reliability improvement; it can be used to easily pass the reliability upper limit achieved by the fault-prevention techniques. Therefore reliability may be improved even further by applying fault-tolerance methodology. So design and implementation of fault-tolerant systems are of great importance.

Faults are expected to occur during system operation (due to designs'

faults and system deterioration). Therefore the challenge is to couple the potential of new technology with fault-tolerant techniques to produce better systems to deal with these failures.

7.2 - Fault-prevention and testing difficulties

In spite of the great effort that has been devoted to developing testing techniques, a number of serious problems exist. For example; Generating Test Pattern, reducing CPU run time, overcoming CPU memory-size limitations, improving Test Coverage . In the Test Application side the difficulties are; immense test data volumes, High Capital/Operating costs of testers, Long tester time required to apply the test, and Diagnostic resolution.

The list of difficulties would be even longer for sequential circuit, redundant circuit, and hazard detection as most test techniques are for combinational circuits.

7.2.1 - Test Pattern Generation

As the use of Automated Test Equipment (ATE) has grown, so has the demand for engineers and programmers to develop the programmes that control these testers. Despite considerable efforts at developing rigorous, systematic approaches to writing test programs for ATE, the test engineer does not currently enjoy computer assistance that approaches the level of the designer's CAD tools.With limited internal access to increasingly complex circuits, the test generation problem is a most untenable one. More efforts in the area of testing resulted in a

new VLSI design methodology called Design For Testability (DFT). The use of testability measures as a design criterion is one particularly successful solution to this "testing problem".Taking this approach the designer tries to increase the controllability and observability of the logic on the chip, but there is a limitation on the number of pins on large chips.

In addition, there are new and complex failures being observed with very large scale integration (VLSI) circuits. These problems are already causing difficulties with the testing of the existing complex chips, and testing is expected to be even more difficult with the higher-complexity chips that are being proposed. When one considers that a complete system consists of many boards, each consisting of many chips, the magnitude of the task is overwhelming.

7.2.2 - Cost of testing

As digital systems become more complex, it is feared that the cost of testing will become a major part of the cost of the system.

7.2.3 - Testing VLSI chips (exhaustive testing)

Testing all embedded elements (exhaustive testing) in a large chip is not practical. As an example exhausting testing of the Motorola 68000 microprocessor would take many years of CPU time.

Test program development is a very difficult task, exhaustive testing of VLSI circuits is not practical, and the use of ATE is very costly.

7.3 - Reliability and Fault-Tolerant computing

Reliability techniques have become of increasing interest for general applications of computers because of several recent trends. A few of these trends are listed below.

Harsher environments : Computer systems have left the clean environments of computer rooms for industrial environments. Temperature and humidity vary widely. The primary power supply may fluctuate, and there may be more electromagnetic interference.

Novice users :The typical user is not sophisticated about the operation of the system.

Increasing repair costs : As hardware costs continue to decline and labour costs escalate, frequent field service calls become much more expensive than adding redundancy to improve system reliability.

Large systems : As systems become larger, there are more components that can fail. Since the overall failure rate is directly related to the failure rates of the individual components, fault-tolerant designs may be required to keep the over-all system failure rate at an acceptable level.

From the point of view of reliability, computing systems can be grouped into four different applications. The applications are ordered by increasingly stringent reliability requirement.

General-Purpose commercial systems which are very susceptible to transient errors (due to close timing margins) and permanent faults (due to their complexity). As performance demands increase, fault tolerance may be the only recourse to building commercial systems

with sufficient mean time to errors (MTTE) to allow useful computation. Occasional errors that disrupt processing for several seconds are tolerable as long as automatic recovery follows. Example of these systems are VAX 11/780, IBM S/360 - S/370 - 4300, and Univac 1100/60.

High availability Systems share resources among many users and the occasional loss of one user is acceptable. In these systems corporation of redundancy techniques to improve their reliability has been very successful. Tandem, Pluribus, ESS, and Intel 432 are examples of this group.

Long - life systems such as unmanned spacecraft can not be manually maintained over the system operating life (frequently 5 or more years). These systems are highly redundant, with enough spares to survive the mission with the required computational power as the peak computational requirement is often at the end of system life. Voyager and STAR computers are examples of long-life systems.

The last application group which places the highest demand on system reliability is called *Critical Applications Systems* . The most stringent requirement for fault-tolerance is in real-time control systems, where faulty computations can jeopardise human life or have high economic impact. In these cases, computation must not only be correct, but recovery time from faults must be minimised. SIFT and FTMP are examples of avionic computers designed to control dynamically unstable aircraft. Their design goal is a failure probability of less than 10^{-9} for a 10 hour mission.

By now it is apparent that the present requirements for highly reliable computation, highly available systems, and long life applications make design and implementation of fault-tolerant systems at great importance. These requirements can not be achieved through the fault prevention techniques *alone* as discussed in this dissertation.

This dissertation has provided three new hardware fault-tolerant techniques. In each of these schemes a majority voter is used to vote on the outputs of the replicated modules. Five channel implementations of these techniques were compared and their behaviours in the presence of multiple failures were examined.

In the first technique when a single module fails, its output to the voter becomes stuck-at-0 and the output of the next module becomes stuck-at-1. When another failure occurs, the system reconfigures itself and the switch forces the output of one of the faulty modules to be s-a-0 and the other to be s-a-1.

In the case of three module failures (if they happen sequentially), the first faulty module becomes s-a-0, and the other two faulty modules become s-a-1, thus the voter computes the correct output to the outside world. Fewer gates were used for the implementation of the switch in this design than in the designs proposed by others for example [SuDuCa80], [SiMc73], and [SoMa78]. Therefore the switch reliability is better than the other switches used in the above mentioned schemes.

The switch and the disagreement detection circuits in the second design developed in this work (Multiple Fault Tolerant

Reconfiguration Structure MFT-RS) were implemented with fewer gates than any other similar designs proposed by others. In this scheme when a single failure occurs, only the output of the faulty module becomes s-a-0 and the other modules' outputs remain unchanged. But in the cases of double and treble faults the system reconfigures as in the first design. One important feature of this design is the structure of the switch component which enables us to model each module and its related switch component as a pair, which is very helpful in reliability evaluation of the system.

During this research a powerful fault-tolerant technique called Highly Reliable-Highly Efficient (HR-HE) was developed and it was shown that this design is more reliable than any other similar fault-tolerant technique.

This design has the same "pair modelling" feature mentioned in the MFT-RS, and can tolerate up to N-2 module failures (where N is the number of modules). In the event of failures, the switch forces the logical outputs of the odd faulty modules in the structure to 1, and the even faulty modules to 0.

In practice this means that every pair of faulty modules cancel each other in the voting mechanism until two fault-free modules remain.

It was shown that the HR-HE structure has a few advantages over the other designs. A five channel HR-HE has advantages over 5MR, hybrid(3,2) [SiMc73], 5MR Reconfiguration Scheme [SuDuCa80], and a 5 channel Sift-out Redundancy Structure [SoMa78] from the point of

view of both the simplicity of their switching structures and the number of faults that can be tolerated (particularly for N>5).

The HR-HE is a very good candidate in "Critical applications", "long-life applications", and "High availability applications".

The structures of the three techniques developed in this work are such that the complexity of their switching mechanisms grow linearly with the number of modules but the voting mechanism complexity increases significantly. This is a better approach than those schemes (e.g. hybrid with iterative switch [SiMc73]) in which the switching complexity increases significantly and the voter's complexity remains constant or grows linearly with the number of modules because it is easier to implement a complex voter than a complex switch (voters have more regular structures).

As the voters in fault-tolerant designs are the most critical components, a regular and simple structure was presented in chapter five and it was shown that a significant reduction in the number of transistors can be achieved compared with other designs.

VLSI technology has many promising applications including the design of super computers that use wafer-scale integration technology. This factor possess the potential of major innovations in computer architecture.

The designs proposed in this work are suitable for WSI architectures, in which a whole system with high fault-tolerance and high reliability is required.

7.4 - Application of Fault-Tolerant techniques in Testing

The detection and location of faults play a significant role in the implementation of fault-tolerant systems. Even in those computer systems that are not labelled as fault tolerant, software or self-test capability is provided to do fault-detection and, perhaps, fault location. System diagnosis is a very important component of the maintenance strategy for conventional (i.e. non fault-tolerant) systems. The maintenance function is a very costly one because it is frequently performed away from a centralised, cost-effective facility and is personnel intensive. Thus design for fault-tolerance, including use of testability features and effective system diagnosis tools, is vital to cost containment. In addition, customer satisfaction is frequently tied to minimising the downtime of the system, which in turn requires rapid fault detection and location.

In fault-tolerant systems, diagnosis is an important tool. Regardless of the fault-tolerance principles that serve as a basis for the system design, it is important to detect and ultimately identify faulty units and take the necessary actions to restore the system to a fault-free condition. Because of the critical computational environments of many such systems, diagnosis is frequently implemented in hardware in order to minimise the time required for corrective action. The disagreement detection circuits used in the proposed designs can play an important role here. The outputs of the detection circuit shows if there is any disagreement between the modules' outputs and the overall system output. A monitoring system therefore can be used to warn the user

when any fault is detected and informs the user that the system is in degradation mode. Then the faulty module will be replaced automatically or manually if it is possible.

These fault-tolerant techniques can also be used for testing modules to detect if they are faulty or not. Suppose that we have a seven channel HR-HE system. Then if the test vectors are applied to this system the disagreement detection circuit detects if there is any mismatch between the modules. Then the monitoring circuit at the outputs of this detection circuit could report which modules do not match with the system output. That is the faulty module can be detected and located and if necessary can be replaced .

7.5 - Reliability Modelling

In this dissertation a new reliability model was developed to study the case when each module and its related switch component can be modelled as a pair for reliability evaluation.

Extensive comparisons were made between the different fault-tolerant techniques (including the new techniques). The result of these analyses were plotted, and the advantages of the new techniques were demonstrated.

The reliability comparisons provide knowledge as to the circumstances under which a given redundancy technique should be used. The final chapter of the dissertation describes how this knowledge can be incorporated in an expert system to form a fault-tolerant advisor as part of a CAD system.

7.6 - Future research

As the scale of integration of circuits continues to increase through advances in semiconductor technology, it seems inevitable that future hardware systems will become ever more complex, and therefore increasingly liable to design faults. The current hardware design testing and techniques seems to be inadequate to handle this problem. Hence effective fault-tolerance for complex hardware systems still requires new and more efficient techniques. Fault-tolerance is achieved by the use of redundancy, but the application of redundancy has two major disadvantages. Firstly it occupies some area on chips that can be used for other components, and secondly increases the complexity of the system. Therefore an accurate and detailed analysis is needed to determine the frequency (or percentage) of failure of each individual component in a system. The result of this analysis will be very helpful in fault-tolerant designs such that it would inform the designers for an efficient use of redundancy in the system.

Further investigation is also necessary to establish a general reliability model which can be used for different types of fault-tolerant techniques. Currently different reliability models are used for different groups or individual techniques.

To be able to have a powerful expert system as part of a CAD tool more extensive knowledge is needed to be gathered to teach the knowledge

base and more sophisticated techniques should be used for the implementation of the system.

REFERENCES:

[AbSi74] J. A. Abraham and D. P. Siewiorek, "An algorithm for the accurate evaluation of triple modular redundancy networks," IEEE Trans. Comput.,vol., C-23, pp. 682- 692, July 1974.

[AnMa67] J. E. Anderson and F. J. Macri, "Multiple redundancy applications in a computer," in Proc. 1967 Annu. Symp. Rel., Washington, D. C., pp. 553-562, 1967.

[Av67] A. Avizienis, "Design of fault-tolerant computers," in 1967 Fall Joint Comput. Conf., AFIPS Conf. Proc., vol. 31. Washington, D.C. Thompson, 1967, pp. 733-743.

[AvEt71] A. Avizienis, et al.,"The Self Testing And Repairing computer (STAR): An investigation of the theory and practice of fault-tolerant computer design," IEEE Trans. Comput., vol. C-20, pp. 1312-1321, 1971.

[BaHa69] M. Ball and F. Hardie, "Majority voter design considerations for TMR computer," Comput.

Design, pp. 100-104, Apr. 1969.

[BoCaJe71] W. G. Bouricius, W. C. Carter, D. C. Jessep, R. P. Schneider, and A. B. Wadia, "Reliability modeling for fault-tolerant computers," IEEE Trans. Comput., vol. C-20, pp. 1306-1311, 1971.

[BoCaRo67] W. G. Bouricius, W. C. Carter, J. P. Roth and P. R. Schneider, "Investigations in the design of an automatically repaired computer," in 1st Annu. IEEE Comput. Conf. Digest, Sept. 1967, pp. 64-67.

[BoLiSe80] L.A. Boone, H.L. Liebergot, and R.M. Sedmak, "Availability, reliability and maintainability aspects of the Sperry Univac 1100/60", Proc. Int. Symp. Fault-tolerant Computing, 3-8 1980.

[CaJeBo70] W. C. Carter, D. C. Jessep, W. G. Bouricius, A. B. Wadia C. E. McCarthy, and F. G. Milligan, "Design techniques for modular architecture for reliable computer systems," IBM Rep. RA 12, Contract NAS8-24883, Mar. 1970.

[CaSc68] W. C. Carter and P. R. Schneider, "Design of dynamically checked computers," in Proc. IFIP Congr. 1968, vol.2, Edinburgh Scotland, Aug. 1968, pp. 878-883.

[ChAv78] L. Chen, and A. Avizienis, "N-version programming: a Fault-tolerant approach to reliability of software operation", Proc. Int. Symp. Fault Tolerant Computing 3-9 1978.

[Fl 58] B. J. Flehinger, "Reliability improvement through redundancy at various system levels," in Conv. Rec. 1958 IRE Nat.Conv., Part6, pp. 137-151; also IBMJ. Res. Develop., Apr. 1958.

[Fe57] W. S. Feller, An Introduction to Probability Theory and Its Applications, vol. 1. New York: Wiley, 1957.

[Fe83] M. Feuer, "VLSI Design Automation: An Introduction", Proceedings of the IEEE, 71 no. 1 Jan 1983, pp. 5-9

[GoGrLe67] J. Goldberg, M. W. Green, K. N. Levitt, and H. S. Stone, "Techniques for the realization of ultra-reliable spaceborne computers," Stanford Res. Inst., Menlo Park, Calif., Interim Sci. Rep. 2, Project 5580, Oct. 1967.

[GoLeSh66] J. Goldberg, K. N. Levitt, and R. A. Short, "Techniques for the realization of ultrareliable spaceborne computers," Stanford Res. Inst., Menlo Park, CA, Final Rep.-Phase I, Project 5580, Sept. 1966.

[GrMiRo62] J. E. Griesmer, R. E. Miller, and J. P. Roth, "The design of digital circuits to eliminate catastrophic failures," in redundancy Techniques for Computing Systems. Washington, D.C. Spartan, 1962, pp. 328-348.

[Ha89] M. Hafezparast, "A Highly Reliable-Highly Efficient Hardware Redundancy Structure", Submmited for the IEEE Trans. Comp. Special Issue on Fault-Tolerant Computing Apr. 1990.

[HaZi89] M. Hafezparast, and R. Zimmer, "Multiple Fault tolerant-Reconfigurable Structure", Submmited for the Journal of Electronic Testing: Theory and Applications (JETTA), 1989.

[HaWaLe] F. Hayes-Roth, D.A. Waterman, D.B. Lenat, "Building Expert Systems", Pub. by Addison-Wesley Pub. Co. Inc 1983.

[HoSmLa78] A.L. Hopkins, T.B. Smith, and J.H. Lala, "FTMP- a highly reliable fault-tolerant multiprocessor for aircraft", Proc. IEEE, 1221-1239 Oct.78.

[IBM80] IBM, "Error Detection Capabilities in Micoprocessors", IBM Technical Disclosure Bulletin Vol.22 No. 10 pp. 4485 - 4487 Mar 80.

[KaEt78] D. Katsuki, et al., "Pluribus-an operational Fault-tolerant multiprocessor", Proc. IEEE, 1146-1157 Oct. 78.

[Kn63] J. K. Knox-Seith, "A redundancy techique for improving the reliability of digital systems,"Stanford Electron. Lab., Stanford Univercity, Stanford,CA, Tech. Rep. 4816-1, Des. 1963.

[Kn64] J. K. Knox-Sieth, "Improving the reliability of digital system by redundancy and restoring organs," Solid-State Electronic Lab., Stanford Univ., Stanford, CA, Tech. Rep. 4816-2, Aug 1964.

[Lo76] J. Losq, "A Highly Efficient Redundancy Scheme: Self-Purging Redundancy", IEEE Trans. on Comput., vol. c-25, no. 6, June 1976.

[LyVa62] R. E. Lyions and W. Vanderkulk, "The use of triple modular redundancy to improve computer reliability," IBM J. Res. Develop. vol. 6, 1962, pp. 200-209.

[Ma69] F. P. Mathur, "Reliability modeling analysis of a dynamic TMR system utilizing standby spares," in Proc. 7th Allerton Conf. Circuit and Syst. Theory, 1969, pp. 243-249.

[Ma70] F. P. Mathur, "Reliability modeling and architecture of ultrareliable fault-tolerant digital computers," Ph.D. dissertation, Dep. Comput. Sci., Univ. Califonia, Los Angeles, Microfilm reorder no. 71-662, June 1970.

[MaAv70] F. P. Mathur and A. Avizienis, "Reliability analysis and architecture of a hybrid redundant digital system: Generalized triple modular redundancy with self-repair," in 1970 Sprin Joint Comput. Conf.,AFIPS Conf. Proc., vol. 36. Washington DC: Thompson,1970, pp. 375-383.

[Ma71] F. P. Mathur, "On reliability modeling and analysis of ultrareliable fault-tolerant digital systems," IEEE Trans. Comput., vol. C-20, pp. 1376-1382, Vov. 1971.

[MaSo75] F. P. Mathur and P. T. deSousa, "Reliability modeling and analysis of general modular redundant systems," IEEE Trans. Reliability, vol. R-24, pp. 296-299, Dec. 1975.

[MeCo80] C. Mead, and L. Conway, " Introduction to VLSI systems" Addisson-Wesley Publishing Company 1980.

[MoSh56] E. F. Moore and C. E. Shannon, "Reliable circuits using less reliable relays," J. Franklin Inst., vol. 262, pt. I, pp. 191-208, and pt. II, 281-297, 1956.

[Mu86] A. Mukherjee, "Intro. to nMOS and CMOS VLSI systems design", Prentice/Hall International, Inc. 86.

[MuIaOk87] J.D. Musa, A. Iannio, K. Okumoto, "Software Reliability, Measurment, Prediction, Application", Mc GrawHill Int. Ed. 87.

[Na87] C. Naylok, "Build your own Expert System", Pub. by Sigma Press, a division of John Wiley & Sons, 1987.

[Ne56] J. von Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components." in Automata Studies, C. E. Shannon and J. McCarthy, Eds. Princeton, N. J.: Princeton Univ. press, 1956, pp. 43-98.

[Ne87] D. Nebendahl, "Expert Systems, Intoduction to the techniques & Applications", Pub. by Siemens Aktiengesellschaft, J. Wiley & Sons, 1987.

[Og74] R. C. Ogus, "Fault-tolerance of the iterative cell array switch for hybrid redundancy," IEEE Trans. Comput., vol. C-23, pp. 667-681 July 1974.

[Og75] R. C. Ogus, "Reliability analysis of hybrid redundancy systems with nonperfect switches," Technical Report 65, Digital Systems Laboratory, Stanford University, Stanford, CA,1974.

[OsJo80] B.E. Ossfeldt, and I. Jonsson, "Recovery and diagnostics in the central control of the AXE switching system", IEEE Trans. Comput., pp. 482-491 Jun. 80.

[PaOCo81] D.T. Patrick, O'Conner, "Practical Reliability Engineering", British Aerospace Dynamics Group, J. Willey and Sons 1981.

[RoBoCa67] J. P. Roth, W. G. Bouricius, W. c. Carter, and P. R. Schneider, "Phase II of an architectural study for a self-repairing computer," SAMSO TR67-106, NOV. 1967.

[Sh68] R. A. Short, "The attainment of reliable digital systems through the use of redundancy A survey, " IEEE Comput. Group News, vol. 2, pp. 2-17, Mar. 1968.

[SiMc73] D. P. Siewiorek and E. J. McCluskey, "An iterative cell switch design for hybrid redundancy," IEEE Trans. Comput., vol.C-22, pp. 290-297, Mar. 1973.

[Si75] D.P. Siewiorek, "Reliability modeling of compensating module failures in majority voted redundancy," IEEE Trans. Comput. (Special Issue on Fault-Tolerant Computing), vol. C-24, pp. 525-533' May 1975.

[SiMc73] D. P. Siewiorek and E. J. McCluskey, "Switch complexity in systems with hybrid redundancy," IEEE Trans. Comput., vol. C-22, pp. 276-282, Mar. 1973.

[SiEt78] D.P. Siewiorek, et al, "A Case study of C.mmp, Cm and C.vmp: part1 - Experiences with fault-tolerance in multiprocessor systems", ibid., 1178-1199 Oct. 78.

[SoMa78] P.T. De Sousa, F.P. Mathur, "Sift-Out Modular Redundancy", IEEE Trans. Comput., Vol. C-27, no. 7, pp. 624-627 Jul. 78.

[SuDuCa] S.Y.H. Su, and E. DuCasse, "A Hardware Redundancy Reconfiguration Scheme for Tolerating Multiple

Module Failures", IEEE Trans. Comput., Vol. C-29, No.3 Mar. 80.

[To64] R. Toeste, "Digital circuit redundancy," IEEE Trans. Reliability, vol. R-13, pp. 42-61, 1964.

[To78] W.N. Toy, "Fault-tolerant design of local ESS processors", Proc. IEEE, 1126-1145 Oct. 78.

[Wa86] D.A.Waterman, "A Guide to Expert Systems", Pub. by Addison-Wesley Pub. Co. 1986.

[We84] S.M. Weiss, and C.A. Kulikowski, "A Practical Guide to designing Expert Systems", Pub. by Rowman & Allanheld Publishers 1984.

[WeEt78] J.H. Wensley, et al., "SIFT: Design and analysis of a Fault-tolerant computer for aircraft control", Proc. IEEE, 1240-1255 Oct. 78.

Appendix A

Equation 4.4 calculates the logic value of α_i (the i^{th} input to the voter in the NMR reconfigurable scheme on page 67).

$$\alpha_i = g_i g_{i-1} \bar{g}_{i-2} + X_i \bar{g}_i + g_i g_{i-2} \bar{g}_{i-1}$$

$$\alpha_1 = g_1 g_7 \bar{g}_3 + X_1 \bar{g}_1 + g_1 g_6 \bar{g}_2$$
$$\alpha_2 = g_2 g_1 \bar{g}_4 + X_2 \bar{g}_2 + g_2 g_7 \bar{g}_3$$
$$\alpha_3 = g_3 g_2 \bar{g}_5 + X_3 \bar{g}_3 + g_3 g_1 \bar{g}_4$$
$$\alpha_4 = g_4 g_3 \bar{g}_6 + X_4 \bar{g}_4 + g_4 g_2 \bar{g}_5$$
$$\alpha_5 = g_5 g_4 \bar{g}_7 + X_5 \bar{g}_5 + g_5 g_3 \bar{g}_6$$
$$\alpha_6 = g_6 g_5 \bar{g}_1 + X_6 \bar{g}_6 + g_6 g_4 \bar{g}_7$$
$$\alpha_7 = g_7 g_6 \bar{g}_2 + X_7 \bar{g}_7 + g_7 g_5 \bar{g}_1$$

In this appendix values of α_i (for i=1 to i=7) will be calculated for two cases as follows:

Logic values of α_i (for i=1 to i=7) have been calculated in the tables below when three or four modules fail. If the i^{th} module fails g_i=1 and $\bar{g}_i = 0$, and if operates correctly $g_i = 0$ and $\bar{g}_i = 1$. The values of α's for any other combination not shown in the table are similar to one of the cases shown in the table.

Case -1

Three modules fail

Faulty modules

	1, 2, 3	1, 2, 4	1, 2, 5	1, 2, 6	1, 3, 5
α_1	0	0	0	0	0
α_2	1	0	1	1	x_2
α_3	1	x_3	x_3	x_3	1
α_4	x_4	1	x_4	x_4	x_4
α_5	x_5	x_5	0	x_5	1
α_6	x_6	x_6	x_6	0	x_6
α_7	x_7	x_7	x_7	x_7	x_7

Case - 2

Four modules fail

Faulty modules

	1, 2, 3, 4	1, 2, 3, 5	1, 2, 4, 5	1, 3, 4, 5	1, 3, 5, 6
α_1	0	0	0	0	1
α_2	0	1	0	x_2	x_2
α_3	1	1	x_3	0	1
α_4	1	x_4	0	1	x_4
α_5	x_5	1	1	1	0
α_6	x_6	x_6	x_6	x_6	0
α_7	x_7	x_7	x_7	x_7	x_7

Appendix B: A Sample Run of the Expert System

```
/*Enter the knowledge into the knowledge-base by the expert*/

/*Type "E" to enter the knowledge*/


(E)nter, (Q)uery, (S)save, (L)oad, e(X)it
choose one: E


        /*Enter the first technique*/

        Fault-tolerant technique: DUPLEX1-WITH 0.98>RSYS>0.99

                /*Enter its attributes*/

                Enter attributes (RETURN to quit)
                        : 2 BASIC MODULES
                        : 0.95>RM>0.96 OR -0.051>YT>-0.105
                        : RSW>0.99
                        : RD>0.99
                        : AREA>2*EACH-MODULE-AREA
                        :


        /*Enter the next technique*/

        Fault-tolerant technique: TMR1-WITH 0.982>RSYS>0.986

                /*Enter its attributes*/

                Enter attributes (RETURN to quit)
                        : 3 BASIC MODULES
                        : 0.95>RM>0.96 OR -0.051>YT>-0.105
                        : VOTER
                        : RVOTER>0.99
                        : AREA>3*EACH-MODULE-AREA
                        :


        /*Enter the next technique*/

        Fault-tolerant technique: 5 CHANNEL HR-HE WITH 0.991>RSYS>0.997

                /*Enter its attributes*/

                Enter attributes (RETURN to quit)
                        : 5 BASIC MODULES
                        : VOTER
                        : RVOTER>0.999
                        : RSWITCH-COM>0.999
                        : 0.80>RM>0.85
                        : AREA>5*EACH-MODULE-AREA
                        :
```

```
/*Type "Q" to query a technique*/

(E)nter, (Q)uery, (S)save, (L)oad, e(X)it
choose one: Q

                /*query process for a TMR*/

                is/does/has it   2 BASIC MODULES?  N
                is/does/has it   3 BASIC MODULES?  Y
                is/does/has it   0.95>RM>0.96 OR -0.051>YT>-0.105?  Y
                is/does/has it   VOTER?  Y
                is/does/has it   RVOTER>0.99?  Y
                is/does/has it   AREA>3*EACH-MODULE-AREA?  Y


                The suitable technique is
                        TMR1-WITH 0.982>RSYS>0.986


(E)nter, (Q)uery, (S)save, (L)oad, e(X)it
choose one: Q

                /*query process for a 7 channel HR-HE*/

                is/does/has it 2 BASIC MODULES?  N
                is/does/has it 3 BASIC MODULES?  N
                is/does/has it 5 BASIC MODULES?  N
                is/does/has it 7 BASIC MODULES?  Y
                is/does/has it 0.80>RM>0.85 OR -0.223>YT>-0.163?  Y
                is/does/has it VOTER?  Y
                is/does/has it RVOTER>0.999?  Y
                is/does/has it RSWITCH-COM>0.999?  Y
                is/does/has it AREA>7*EACH-MODULE-AREA?  Y


                The suitable technique is
                         7 CHANNEL HR-HE WITH RSYS>0.999


(E)nter, (Q)uery, (S)save, (L)oad, e(X)it
choose one: Q

                /*query for a 5 channel HR-HE*/

                is/does/has it 2 BASIC MODULES?  N
                is/does/has it 3 BASIC MODULES?  N
                is/does/has it 5 BASIC MODULES?  Y
                is/does/has it VOTER?  Y
                is/does/has it RVOTER>0.999?  Y
                is/does/has it RSWITCH-COM>0.999?  Y
                is/does/has it 0.80>RM>0.85?  Y
                is/does/has it AREA>5*EACH-MODULE-AREA?  Y


                The suitable technique is
                        - CHANNEL HR-HE WITH 0.991>RSYS>0.997
```

```
        /*Enter the next technique*/

        Fault-tolerant technique: 7 CHANNEL HR-HE WITH RSYS>0.999

                /*Enter its attributes*/

                Enter attributes (RETURN to quit)
                        : 7 BASIC MODULES
                        : 0.80>RM>0.85 OR -0.223>YT>-0.163
                        : VOTER
                        : RVOTER>0.999
                        : RSWITCH-COM>0.999
                        : AREA>7*EACH-MODULE-AREA
                        :


/*Saving the knowledge into the knowledge-base*/

/*Type "S" to save the k-base*/

(E)nter, (Q)uery, (S)save, (L)oad, e(X)it
choose one:choose one: S

                saving knowledge base

                /*knowledge-base is saved*/


/*To do query about a technique*/
/*knowledge-base should be loaded first*/


/*Type "L" to load the Knowledge-base */

(E)nter, (Q)uery, (S)save, (L)oad, e(X)it
choose one:choose one: L

                loading knowledge base

                /*knowledge-base is loaded*/
```

```
(E)nter, (Q)uery, (S)save, (L)oad, e(X)it
choose one: Q

                /*query for a 7 channel HR-HE*/
                /*The system give reasons of selecting a technique*/
                /* if type "W" to a question*/

                is/does/has it 2 BASIC MODULES?  N
                is/does/has it 3 BASIC MODULES?  N
                is/does/has it 5 BASIC MODULES?  N
                is/does/has it 7 BASIC MODULES?  Y
                is/does/has it 0.80>RM>0.85 OR -0.223>YT>-0.163?  Y
                is/does/has it VOTER?  Y
                is/does/has it RVOTER>0.999?  Y
                is/does/has it RSWITCH-COM>0.999?  Y


        /*reasons to select the particular technique*/

        Trying 7 CHANNEL HR-HE WITH RSYS>0.999


                it is/has/does:

                        7 BASIC MODULES
                        0.80>RM>0.85 OR -0.223>YT>-0.163
                        VOTER
                        RVOTER>0.999

                and is/has/does not:

                        2 BASIC MODULES
                        3 BASIC MODULES
                        5 BASIC MODULES

        /*The end of reasonning*/


                is/does/has it RSWITCH-COM>0.999?  Y
                is/does/has it AREA>7*EACH-MODULE-AREA?  Y


                The suitable technique is
                        7 CHANNEL HR-HE WITH RSYS>0.999


/*Type  "X" to exit the system*/


(E)nter, (Q)uery, (S)save, (L)oad, e(X)it
choose one:  X
```