



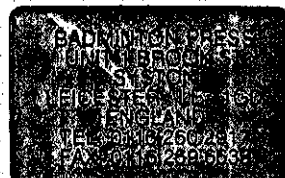
**Pilkington Library**

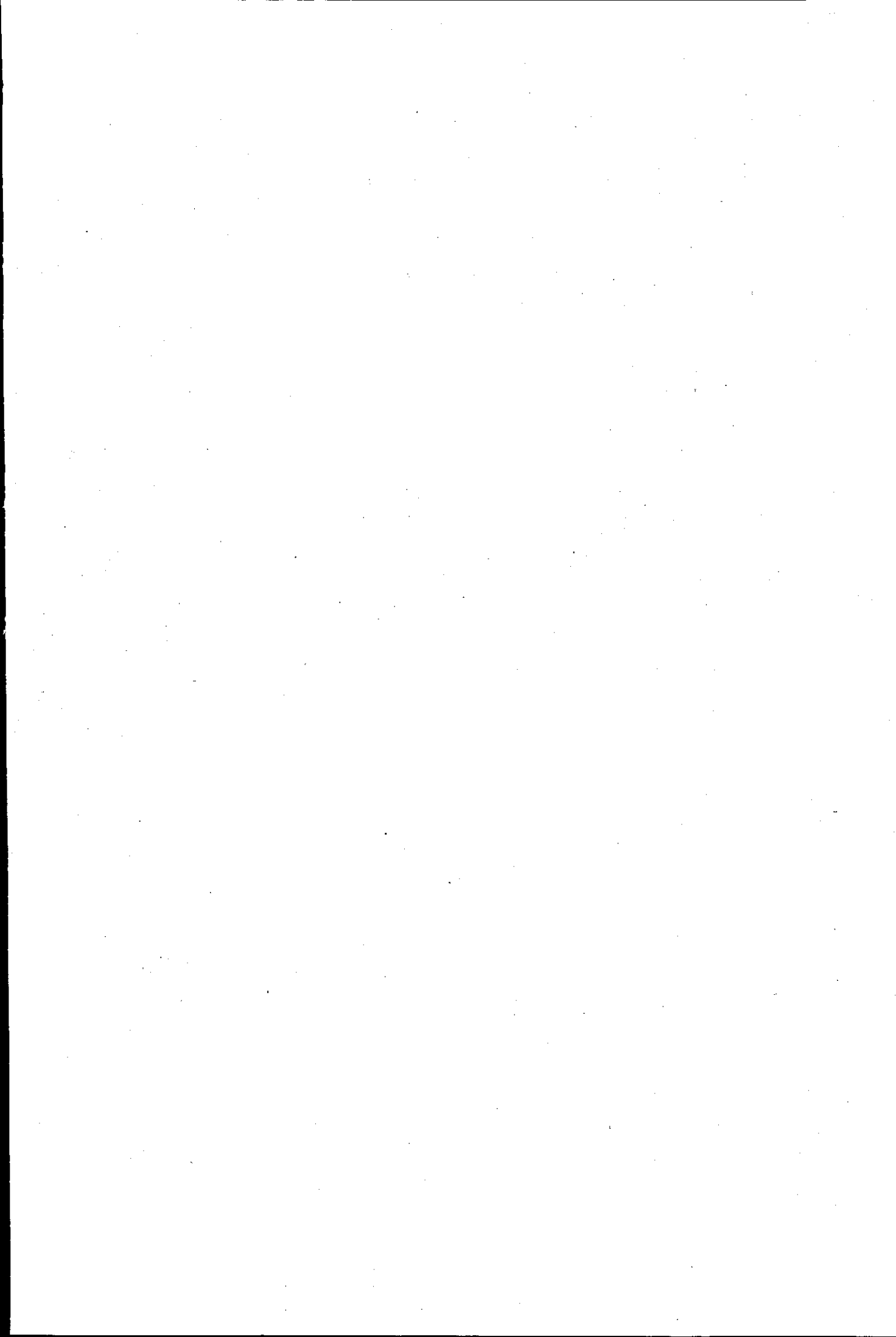
Author/Filing Title ..... YEANDEL, J.

Accession/Copy No. 040160818

Vol. No. ....	Class Mark .....	
25 JUN 1999	LJAN copy	

0401608182





**INVESTIGATIONS INTO THE  
FEASIBILITY OF AN ON-LINE TEST  
METHODOLOGY**

By

**Julian Yeandel  
MSc. AMIEE**


A Doctoral Thesis.

Submitted in partial fulfilment of the requirements  
for the award of

**Doctor of Philosophy  
of  
Loughborough University**

1998

© by Julian Yeandel

 Loughborough University P... ..ary
Date May 98
Class
Acc No 040160818

K0628974

## ABSTRACT

This thesis aims to understand how information coding and the protocol that it supports can affect the characteristics of electronic circuits. More specifically, it investigates an on-line test methodology called IFIS (If it Fails It Stops) and its impact on the design, implementation and subsequent characteristics of circuits intended for application specific IC (ASIC) technology.

The first study investigates the influences of information coding and protocol on the characteristics of IFIS systems. The second study investigates methods of circuit design applicable to IFIS cells and identifies the technique possessing the characteristics most suitable for on-line testing. The third study investigates the characteristics of a 'real-life' commercial UART re-engineered using the techniques resulting from the previous two studies. The final study investigates the effects of the halting properties endowed by the protocol on failure diagnosis within IFIS systems.

The outcome of this work is an identification and characterisation of the factors that influence behaviour, implementation costs and the ability to test and diagnose IFIS designs.

## ACKNOWLEDGEMENTS

Firstly, I would like to thank the Royal Academy of Engineering for providing the necessary funding to attend international conferences.

Secondly, special thanks are more than deserved by my supervisor and mentor Professor Simon Jones with whom I shared numerous stimulating conversations. I would also like to thank him for the patience and encouragement extended by him throughout the duration of the learning process.

Thirdly, I must extend my thanks to members past and present of the Electronic Systems Design Group at Loughborough University for taking the time to listen during the process of idea formulation. In particular I should thank Dr Dave Thulborn who defined the state transition sequence for the UART controller and coded it in VHDL according to the structure defined in chapter 6. Furthermore, he coded the controller model using the 'C' programming language according to my specification.

I also owe my gratitude to Mr Ryan Lim for translating my specification for a software interface to Labview and turning it into reality.

Finally, I shall always be indebted to Daniela and Jeremy, my wife and small son, for their continued support and understanding throughout the last three years. This work would have been impossible without them.

# TABLE OF CONTENTS

## CHAPTER ONE INTRODUCTION

1.1 Introduction and Motivation.....	1
1.2 On-line testing.....	2
1.2.1 Problems addressed by on-line testing.....	3
1.2.2 Advantages and disadvantages of on-line testing.....	5
1.3 IFIS.....	6
1.4 Aims of Thesis.....	9
1.5 Structure of Thesis.....	9

## CHAPTER TWO REVIEW

2.1 Objectives of Chapter.....	11
2.2 Motivation.....	11
2.2.1 Typical causes of system failure.....	12
2.2.2 IFIS specific requirements.....	12
2.3 Hardware Redundancy.....	13
2.3.1 Duplication with comparison.....	13
2.3.2 Fault-secure networks.....	14
2.3.3 N-modular redundancy.....	14
2.3.4 Summary of hardware redundancy techniques.....	15
2.4 Information Redundancy.....	16
2.4.1 Parity codes.....	16
2.4.2 Arithmetic codes.....	17
2.4.3 Unidirectional error detecting codes.....	19
2.4.4 Summary of information redundancy techniques.....	23
2.5 Time Redundancy.....	26
2.5.1 Functional Re-Mapping.....	26
2.5.2 Functional Partitioning.....	27
2.5.3 Duality.....	28
2.5.4 Summary of Time Redundancy Techniques.....	28
2.6 Analysis of Review.....	30
2.6.1 Coding between computation elements.....	30
2.6.2 Design techniques for computation.....	32
2.6.3 Areas open to research within the context of IFIS.....	33

## CHAPTER THREE OVERVIEW OF INVESTIGATIONS

3.1 Objectives of Chapter.....	34
3.2 Objectives of Research.....	34

3.2.1 Identification of Research Topics.....	34
3.2.2 Statement of Research Objectives.....	35
3.3 Status of IFIS Methodology.....	35
3.4 Introduction to Investigations.....	36
3.4.1 Choosing a Version of IFIS.....	36
3.4.2 IFIS Cell Design.....	38
3.4.3 Feasibility study.....	38
3.4.4 Fault Diagnosis in IFIS.....	39

## **CHAPTER FOUR**

### **SELECTING A VERSION OF IFIS**

4.1 Objectives of Chapter.....	41
4.2 Introduction.....	41
4.3 Coding.....	43
4.3.1 Return to zero coding.....	44
4.3.2 Saturated coding.....	45
4.3.3 Application of codes to IFIS.....	45
4.4 Protocol.....	47
4.4.1 Elastic pipelines.....	47
4.4.2 Inelastic pipelines.....	50
4.4.3 Application of protocol to IFIS.....	51
4.5 Behaviour of alternative IFIS versions.....	51
4.5.1 Existing protocol conditions.....	52
4.5.2 Proposed protocol conditions.....	56
4.5.3 Summary of behaviour of alternative versions of IFIS.....	60
4.6 Test Coverage associated with alternative versions of IFIS.....	61
4.7 Conclusions.....	64

## **CHAPTER FIVE**

### **IFIS CELL DESIGN**

5.1 Objectives of Chapter.....	66
5.2 Introduction.....	66
5.3 Cell Partitioning.....	67
5.4 Computation.....	68
5.4.1 Adapting conventional techniques to IFIS.....	68
5.4.2 IFIS code generation: the need for redundancy.....	69
5.4.3 Relevant design properties.....	70
5.4.4 Information redundancy.....	71
5.4.5 Hardware redundancy.....	73
5.4.6 Time redundancy.....	75
5.4.7 Summary of computation techniques.....	77
5.5 Transmission.....	78
5.5.1 Partitioned design.....	79
5.5.2 Non-partitioned design.....	86
5.5.3 Summary of transmission control techniques.....	88
5.6 Conclusions.....	91



## **CHAPTER SIX**

### **FEASIBILITY STUDY**

6.1 Objectives of Chapter.....	93
6.2 IFIS UART Design and Implementation.....	93
6.2.1 Receiver.....	94
6.2.2 Transmitter.....	95
6.2.3 Controller.....	95
6.2.4 Provision for IFIS.....	96
6.2.5 Design issues.....	96
6.3 Results.....	101
6.3.1 Faulty and fault-free operation.....	102
6.3.2 Complexity and performance.....	104
6.3.3 Fault coverage.....	105
6.4 Summary.....	107
6.5 Conclusions.....	107

## **CHAPTER SEVEN**

### **FAULT DIAGNOSIS IN IFIS**

7.1 Objectives of Chapter.....	109
7.2 Motivation.....	109
7.3 Expectations of Failure Diagnosis in IFIS .....	111
7.3.1 Inherent influences of IFIS.....	111
7.3.2 Limitations on resolution using halting sequence.....	113
7.3.3 Summary of expectations.....	118
7.4 Application to IFIS.....	118
7.4.1 Fault Sites Within IFIS Designs.....	119
7.4.2 The Influence of Different Connection Faults.....	120
7.4.3 The Halting Dictionary Generation Algorithm.....	121
7.5 Fault Injection in the IFIS UART.....	124
7.6 Results.....	126
7.7 Conclusions.....	128

## **CHAPTER EIGHT**

### **CONCLUSIONS**

8.1 Objectives of Chapter.....	130
8.2 Review of Objectives.....	130
8.3 Main Conclusions.....	131
8.4 Measures of Success.....	133
8.5 Limitations.....	134
8.6 Further Work.....	136
8.7 Summary.....	137

<b>REFERENCES.....</b>	<b>138</b>
------------------------	------------

<b>PUBLICATIONS.....</b>	<b>146</b>
--------------------------	------------

## LIST OF TABLES

Table 2-1 Properties associated with hardware redundant techniques.....	15
Table 2-2 Properties of information redundant techniques.....	25
Table 2-3 Properties of time redundant techniques.....	29
Table 2-4 Properties of codes suitable for communication.....	31
Table 2-5 Properties of computation techniques applicable to any digital function...	33
Table 4-1 Dual-rail codes and their meanings.....	46
Table 4-2 Code classification.....	53
Table 4-3 Behaviour of the protocol/coding combinations.....	60
Table 4-4 Characteristics of existing protocol conditions on a single data bit RTZ code.....	61
Table 4-5 Characteristics of code/protocol combinations carrying d data bits.....	62
Table 5-1 Complexity of an unprotected architecture for generating IFIS codes.....	71
Table 5-2 Complexity of an architecture protected by information redundancy.....	73
Table 5-3 Complexity of an architecture protected by hardware redundancy.....	74
Table 5-4 Complexity of an architecture protected by time redundancy.....	76
Table 5-5 Suitability of different architectural design approaches to the IFIS function.....	77
Table 5-6 Complexity of alternative checkers.....	82
Table 5-7 Maximum SSA test coverage for alternative checkers.....	84
Table 5-8 Characteristics of the selector units.....	86
Table 5-9 Characteristics of the combined checker/selector architectures.....	88
Table 6-1 Signal names and descriptions.....	102
Table 6-2 Complexities of the three top level UART design blocks.....	104
Table 6-3 Performance of the three top level UART design blocks.....	104
Table 7-1 Classification of activities for different methods of fault diagnosis.....	110
Table 7-2 Classification of activities for different methods of on-line testing.....	110
Table 7-3 Observable primary halting sequence of the system shown in Figure 7-4 mapped to the cells which initiate each halt.....	117
Table 7-4 Application of the X1 model to describe different faults.....	121
Table 7-5 Characteristics of application of the different algorithms to the IFIS UART design.....	122
Table 7-6 Mapping of signal pin names to faults as shown in Figure 7-5.....	125
Table 7-7 The fault code placed on the fault injection channels and the code description.....	126
Table 7-8 Diagnosis of the injected faults.....	127

## LIST OF FIGURES

Figure 1-1	The structure of IFIS systems.....	7
Figure 2-1	Bose-Lin codes.....	21
Figure 2-2	Bose code example.....	22
Figure 2-3	Blaum code example.....	23
Figure 2-4	Recomputing with shifted operands.....	27
Figure 4-1	State transition diagrams for Binary and Return-to-zero systems.....	44
Figure 4-2	State transition diagram for saturated coding scheme.....	45
Figure 4-3	Interconnection of processing elements 'A', 'B' and 'C' forming a pipeline.....	48
Figure 4-4	Data movement in an elastic pipeline.....	49
Figure 4-5	Data movement in an inelastic pipeline.....	50
Figure 4-6	Neighbouring elements within an IFIS pipeline.....	52
Figure 4-7	Architecture used to demonstrate fault conditions.....	54
Figure 4-8	Effects of a specific fault on the behaviour base on the existing inelastic protocol.....	55
Figure 4-9	Interconnection of processing elements according to the proposed protocol.....	57
Figure 4-10	Behaviour of the proposed protocol under fault-free conditions.....	58
Figure 4-11	Behaviour of the proposed protocol under a single injected fault condition.....	59
Figure 4-12	Proposed protocol under fault-conditions: code translators ignoring halt.	59
Figure 4-13	Proportion of states that allow processing in different code/protocol combinations.....	62
Figure 4-14	Proportion of binary state codes that allow processing in different code/protocol combinations.....	63
Figure 5-1	Anatomy of an IFIS cell.....	67
Figure 5-2	Code generation without computation protection.....	70
Figure 5-3	Code generation with computation protection provided by information redundancy.....	72
Figure 5-4	Code generation with computation protection provided by hardware redundancy.....	74
Figure 5-5	Code generation with computation protection provided by time redundancy.....	75
Figure 5-6	Complexity comparison of the techniques applied to computation block design.....	77
Figure 5-7	Protocol validation architectures.....	79
Figure 5-8	Generic decoder architecture.....	81
Figure 5-9	Test properties of the decoder architecture where $n$ is equal to 1.....	83
Figure 5-10	Selector architectures for selecting one of two busses.....	85

Figure 5-11 Combined decoder/selector architecture.....	87
Figure 5-12 Combined comparator/selector architecture.....	87
Figure 5-13 Scaling of design complexity for the alternative architectures.....	89
Figure 5-14 Maximum production test coverage.....	89
Figure 5-15 Maximum test coverage when restricted to protocol obeying vectors.....	90
Figure 5-16 Maximum test coverage when restricted to protocol obeying vectors followed by an enforced freeze.....	90
Figure 6-1 Top level architecture of the UART.....	94
Figure 6-2 IFIS computation architecture.....	97
Figure 6-3 The flow used to generate verification patterns for the UARTs.....	99
Figure 6-4 Design verification in the software and hardware environments.....	100
Figure 6-5 The injected fault, marked by $\alpha$ , activated in the controller.....	101
Figure 6-6 The IFIS UART under normal operation.....	102
Figure 6-7 The IFIS UART under an injected fault condition.....	103
Figure 6-8 Comparison of SSA fault coverage.....	106
Figure 7-1 An IFIS chain where only one halting path exists from the primary input cell to the primary output cell.....	114
Figure 7-2 Generic IFIS system where two paths exist between cell A and cell Z....	115
Figure 7-3 An example IFIS system where two paths of equal length exist between the primary input cell and the primary output cell.....	116
Figure 7-4 An example IFIS system where two paths of unequal length exist between the primary input cell and the primary output cell.....	116
Figure 7-5 Protocol affecting fault sites.....	119
Figure 7-6 The context of each type of fault.....	120
Figure 7-7 The IFIS structure of the IFIS UART.....	124
Figure 7-8 Fault injection sites within the selected IFIS cell contained in the IFIS UART.....	125
Figure 7-9 The relation between fault types which are indistinguishable when halting of primary outputs is used for diagnosis.....	128

# CHAPTER ONE

## INTRODUCTION

### 1.1 INTRODUCTION AND MOTIVATION

The ability to make increasingly complex integrated circuits (ICs) stems from continuing advances in microelectronic technology. Recent surveys showed that in 1970 there were typically a thousand transistors per IC, whereas in 1990 there were a million. This trend is expected to continue resulting in complexities of approximately one thousand million transistors by 2010 [Sherwani95, Meindl87]. Increasing integration results in:

- A reduction in the average costs per logic gate. This is due to reduced packaging costs, a reduction in the numbers of ICs required to implement any specific system and the associated simplification of printed circuit boards.
- An increase in reliability for any specific system if fewer ICs are needed to realise it. The need for fewer ICs results in fewer interconnections between them and therefore increases system reliability [Brasington95, Siewiorek92].

There are a number of problems associated with increased levels of integration, namely:

- Larger ICs are more prone to errors. Technological advances have enabled both the shrinking of minimum geometry and expansions in die sizes. This increase in monolithic area raises the chances of manufacturing defects and increases the susceptibility of ICs to environmental factors.
- Systems containing ICs are more difficult to test. This is due to increased circuit complexity and reduced access to internal circuit nodes.
- The increased susceptibility of ICs to environmental factors suggests that they must be checked more often if their integrity is to be relied on.

Applications are demanding higher levels of integration and higher levels of confidence in system integrity. Many techniques exist that facilitate checking of systems and the ICs that they contain. These techniques can be classified as:

- Off-line testing techniques
- On-line testing techniques

Off-line testing techniques predominate and indeed substantial investment has been made in the development of such techniques. Off-line testing can be used at the time of system (IC) manufacture with the aim of prohibiting the shipment of faulty products. Enormous efforts continue to be devoted to the development of off-line testing techniques in an attempt to reduce test costs while maintaining product quality.

Despite the enormous attention afforded to the development of off-line testing techniques, the pace of such developments is not keeping up with technological advancements [Bahram92]. This coupled with relatively high costs associated with production testing makes it increasingly difficult to test for all possible defects at the time of fabrication [Dislis95, Thompson96]. It is therefore more likely that faulty products are put into service.

## 1.2 ON-LINE TESTING

On-line testing techniques occur concurrently with normal operation and thus they provide up-to-date information without the need to set aside time for testing. The objectives of the on-line testing of electronic designs are twofold, namely:

- To detect sources of failure that were caused during system manufacture but were not detected prior to commissioning: either those that cause logical faults or are responsible for premature ageing (for example, high resistance bridging faults or undersized metal tracks).
- To detect errors that are caused after system commissioning. These may be

caused by environmental conditions such as heat, vibration, alpha particles, user error or abuse. Intermittent and transient faults are 20 - 100 times more prevalent in commissioned systems than those directly attributable to premature ageing [Siewiorek92].

The testing of commissioned digital electronic systems should provide facilities for detecting abnormalities in system behaviour due to environmental conditions and wear-out in addition to compensating for the inadequacies of production testing.

### **1.2.1 Problems Faced by On-line Testing**

The problems faced by on-line testing of electronic systems include:

- The difficulty in ensuring system integrity at the time of manufacture combined with environmental factors makes it difficult to predict the nature of faulty behaviour.
- It is difficult to differentiate between faulty operation and fault-free operation at any point in time because reference information is not available. Unlike the production (off-line) test environment, system stimuli are not pre-determined, and consequently system responses cannot be predicted prior to testing.
- It cannot be assumed that correct behaviour results from structural integrity. Faults caused by the operating environment may not be permanent but their effects may be long lasting. Intermittent and transient faults may corrupt the state of a system at a specific point in time. The fault may disappear leaving a physically fault free-system that subsequently fails.
- Differentiating faulty behaviour from fault-free behaviour for any error that may occur during the lifetime of a specific system requires knowledge of the errors that may occur.
- Increased system complexity makes immediate fault detection more difficult. The effects of an internal failure that causes a change in the state graph of a system may not have an immediate effect on primary system outputs.

On-line test techniques approach these problems by allowing additional tasks to be performed concurrently with normal operations, namely:

- The generation of additional data that can be used as a dynamic reference.
- The dynamic checking of operational data against that reference data.

Methods of providing reference information include:

- Hardware redundancy
- Information redundancy
- Time redundancy

Often it is possible to place specific techniques under more than one redundancy category. In most cases, additional hardware is required even if the specific technique relies on re-calculating a function output during consecutive periods of time (time redundancy) or relies on the comparison of different representations of the same data (information redundancy).

Generally, hardware redundant techniques exploit the properties associated directly with circuit structure to ensure the independence of operational and reference data. These techniques are typically applied to facilitate the checking of operational data calculation.

Information redundant techniques exploit the dynamic relationship between operational data and reference data to facilitate checking. By knowing the relationship between incoming data and supplementary (reference) information, the relationship between outgoing operational and reference data can be predicted. Information redundancy is therefore used to facilitate the checking of both data calculation and communication.

Time redundant techniques sacrifice data rate to facilitate on-line testing. These techniques can be applied to calculation, by comparison of the results of successive



function computation or to communication by return transmission according to a specific protocol [Grant89].

### 1.2.2 Advantages and Disadvantages of On-line Testing

On-line testing has a number of advantages and disadvantages associated with it. The advantages include:

- On-line testing enables immediate detection of errors that could result in system failure. This is important for applications where loss of life or security could result under prolonged failure conditions. Permanent faults and temporary faults are automatically flagged as soon as the system detects anomalous behaviour.
- On-line testable designs can reduce the time taken for production testing because they incorporate response analysis facilities [Gupta96]. This reduces communication between the device under test and the tester while also permitting higher clock frequencies to be used during testing.
- While the fault-models on which on-line testing techniques are based do not reflect the defects common to CMOS ICs, on-line testing detects errors that arise from them as soon as digital values are affected.
- On-line testing increases system availability because it removes the need for periodic off-line testing.

The disadvantages associated with on-line testing include:

- Additional action(s) must be taken before the results of on-line testing are to be of use. It is pointless to know that system integrity has been compromised unless this information results in action. This action may be undertaken by circuitry that itself is error prone.
- Additional system outputs are required to represent the system error status.
- The degree of fault coverage achieved during any specific period of time is not directly quantifiable during normal operation. This is because no specific

restrictions are exercised over incoming data.

- A complexity penalty is always incurred in comparison with the non-test oriented design. Depending on the technique employed, this penalty can be in excess of 100% for error detection. [Al-Saad96, Siewiorek92]
- A greater complexity penalty is typically incurred to obtain a specific fault coverage in the on-line testing environment than for an equivalent design in the off-line environment. This is at least partly due to the additional complexity incurred by the data analysis hardware incorporated within on-line testable designs.

Clearly, there are a number of disadvantages associated with on-line testing. If these disadvantages are to be combated, new techniques must be explored and their properties characterised.

### 1.3 IFIS

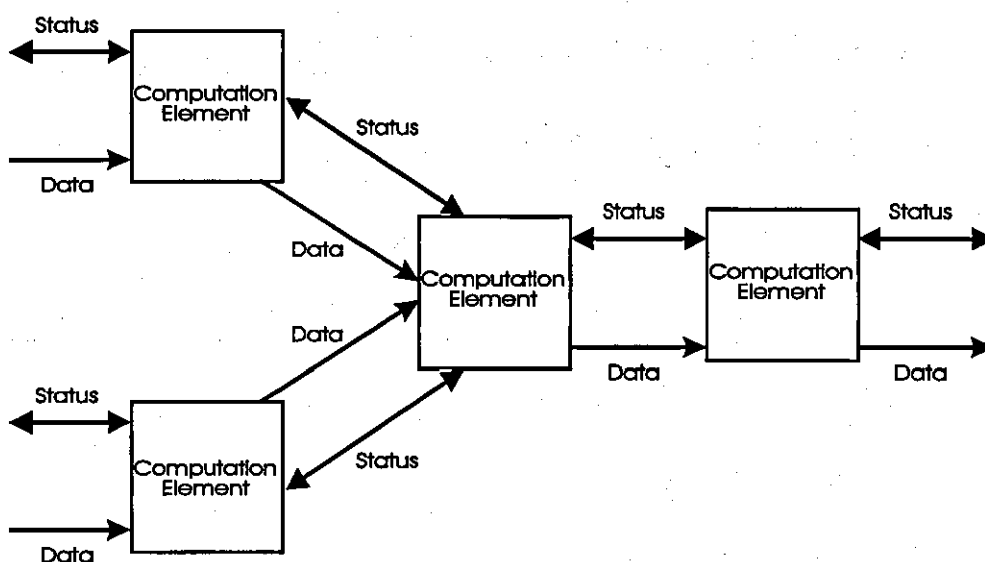
IFIS (If it Fails It Stops) is an on-line design for testability approach that is currently under investigation at Loughborough University. IFIS performs error management in addition to error detection. If knowledge of system integrity is to serve some purpose then some action must be undertaken following error detection. This action may be performed by a dedicated error manager or by other means. When performed by a dedicated error manager, communication between it and the error detection mechanism must exist. Both the error manager and communication link may themselves be error prone. Error management is inherent in IFIS systems and therefore removes these additional sources of potentially unchecked error. The additional advantages of IFIS are summarised below:

- The effect of a failure is propagated through to all circuit elements causing them all to halt. Thus, the system error status can be discovered by observing any primary output.
- The need to generate and apply test vectors to propagate failures to an

observable primary output (at the time of manufacture) is eliminated. Faults within IFIS cells need only be sensitised to the cell output because propagation to a system output is automatic by virtue of the halting mechanism.

- The halting mechanism used to flag errors and manage the flow of erroneous data is not maskable and does not permit the propagation of corrupt data.

Figure 1-1 shows a structure that incorporates communications supporting the above description. This is the generic structure of IFIS systems. Figure 1-1 shows that IFIS systems comprise computation elements (cells) and interconnections between cells.



*Figure 1-1 The structure of IFIS systems.*

Computation elements interpret received status information, compute some function of the received data and generate status information. The interconnections carry data and status information between computation elements. Data flows between computation elements in a conventional manner while status is propagated bi-directionally to neighbouring elements.

This type of design structure is normally associated with asynchronous digital systems where data is permitted to progress from its source depending on the readiness of its destination to receive it [Hulgaard94,David92,Sutherland89]. The destination cell signals its readiness to receive new data only after it has processed old data. In

practical asynchronous designs this signal is combined with the recently computed data by using information redundancy in the form of data coding [Linder96, Dean91].

In contrast to the data coding found in asynchronous designs, the coding found in IFIS designs uses information redundancy to incorporate the error status of IFIS cells [Jones91]. A cells error status reflects the integrity of incoming data and the status of neighbouring cells.

Following the detection of an error by a computation element, the error status of the element is updated and the information passed to neighbouring elements. The neighbouring elements respond by refusing to compute and by updating their associated error status accordingly. A halting of data processing results, progressing away from the error source until all primary system inputs and outputs are reached. This mechanism provides a clear error indication that cannot be ignored.

From the behavioural description of IFIS, it is apparent that the integrity of incoming data and status information must be checked by each computation element before its local error status can be calculated. Incoming data and status information are carried from neighbouring elements via interconnections that may themselves be prone to errors. This suggests that reference information must accompany the data/status information to enable validation to be performed.

A variety of communication coding techniques exist from which a suitable code must be selected to enable appropriate error detecting capabilities to be endowed while still permitting efficient data transfer. Included in the list of candidate codes is the dual-rail return-to-zero code presented in the original IFIS patent [Jones91].

The selection of reference information depends on the severity of the error to be detected, and the type of error that is most likely to occur. Additionally, the domains of computation and communication are subject to different types of errors. For this reason, a method must be devised to additionally ensure the integrity of the circuit used to perform data computation and communication code generation.

From the description of halt propagation within IFIS systems, it can be deduced that an interesting consequence of this mechanism exists, namely: the halting sequence of primary system outputs must contain information relating to the source of error. The utility of this information is yet to be determined.

#### **1.4 AIMS OF THESIS**

This thesis focuses on the IFIS on-line test methodology. More specifically, this thesis provides an assessment of different aspects of the methodology and offers evidence as to its suitability for application to on-line testable systems. This evaluation is achieved through the fulfilment of the following aims:

- To identify protocol/coding schemes that provide the halting mechanism characteristic of IFIS systems and to assess their impact on system level test coverage, complexity and performance.
- To discover the utility of the halting sequence information available at primary outputs for locating the source of error.

#### **1.5 STRUCTURE OF THESIS**

This chapter has introduced and justified the need to provide on-line testing. Furthermore, the IFIS on-line test methodology has been presented and the aims of this thesis have been identified.

Chapter 2 supports the aims of this thesis by reviewing design techniques relevant to IFIS. The requirements and expectations of IFIS systems are presented and the suitability of selected state-of-the-art techniques is assessed within this context. This assessment results in an identification of areas open to research.

Chapter 3 formulates a set of objectives to meet the research aims. Furthermore,

investigations are introduced that target the fulfilment of the formulated objectives.

Chapter 4 describes an investigation that results in the selection of a protocol/coding combination for future IFIS systems. The candidate IFIS protocols and data codes are described and the suitability of their characteristics to a design for test environment are assessed.

Chapter 5 describes an investigation that assesses the influence of using specific structural design techniques selected from those described in Chapter 2 on the properties associated with IFIS cells: the fundamental components within an IFIS system. The investigation results in the determination of a generic cell structure for future IFIS designs.

The investigation described in Chapter 6 builds on the results of the investigations described in Chapter 4 and Chapter 5. A complex design is realised using IFIS and implemented on FPGA. The resultant design is used to demonstrate the behaviour of an IFIS system under both faulty and fault-free conditions. The characteristics of the system are presented.

Chapter 7 establishes the inherent support of failure diagnosis provided by IFIS designs. It also describes the limitations imposed on the resolution attainable using the halting property of IFIS designs as a means of diagnosis. Furthermore, Chapter 7 describes an algorithm which can be used to perform fault diagnosis and introduces extensions to the original IFIS cell model with the goal of discovering the resolution implications of model extension. The algorithm is then applied to responses of the case study implementation under controlled fault conditions.

Chapter 8 summarises and concludes this thesis, discussing the measure of success achieved with respect to this work.

## **CHAPTER TWO**

### **REVIEW**

#### **2.1 OBJECTIVES OF CHAPTER**

The main objectives of this chapter are to:

- Introduce the requirements imposed by the IFIS on-line test methodology.
- Present and compare different state-of-the-art design techniques used for on-line testing.
- Assess the suitability of selected techniques to IFIS identifying shortcomings and areas warranting investigation.

#### **2.2 MOTIVATION**

IFIS is a developing test methodology that uses encoded data and handshaking between computation elements to achieve system level on-line test [Jones91].

For any on-line test methodology to be widely applicable, it should result in designs that conform to the following requirements:

- Same data throughput/data-flow under error-free conditions as equivalent conventional systems. This minimises the impact of incorporating on-line test.
- Ability to detect a high proportion of possible system failures (error coverage).
- Minimal delay between error detection and error flagging (error latency).
- Minimal hardware complexity penalty.

Furthermore, to avoid unnecessary engineering effort, while exhibiting useful properties, it is important that the on-line test methodology be tailored to detect and flag errors typical of the intended operational environment.

### 2.2.1 Typical Causes of System Failure

IFIS is intended for implementing systems comprising ICs and electrical interconnections. The type of errors to be detected by IFIS result from previously undetected manufacturing defects and environmental conditions.

The faults most commonly attributed to CMOS IC manufacture are those resulting from missing connections (stuck-open faults) and extra connections (bridging faults) [Chakra94,Aitken92,Ferguson91,Champac91,Nigh90,Mao90]. Such faults often result in unintended sequential behaviour and are not restricted to specific IC structures.

Under operational conditions, ICs are particularly prone to unidirectional errors. Unidirectional errors are characterised by ones being transformed to zero's ( $1 \rightarrow 0$ ) or vice-versa, but not both at the same. Furthermore, the direction of error cannot be predicted a priori. Unidirectional errors are typically associated with ROMs, RAMs and Bus Structures [Pradhan80a,Blaum88,Piestrak95]. Current levels of integration are characterised by the feature that signal routing area is significant and therefore stands a significant chance of being a source of error [Brassington95,Sherwani95].

### 2.2.2 IFIS Specific Requirements

IFIS imposes a specific structure to support its halting mechanism. This mechanism, coupled with the original specification impose two further restrictions, namely:

- The structure of computation elements should not inherently exclude the computation of particular functions. This is to enable the IFIS methodology to be applicable to any digital system.
- The coding used for inter-cell communications should be chosen to facilitate modular checker design. One function performed by IFIS cells is code checking. Incoming codes represent data and status information from multiple sources where no restriction is applied to bus width. Checker architecture should support the design of different checkers accepting different numbers of codes, perhaps of different code widths.



As an on-line test methodology providing protection of computation and communication, IFIS should be designed using appropriate on-line testing techniques. Computation can be protected by hardware redundancy, information redundancy and time redundancy techniques, while communication requires information redundancy that must itself, be computed. To identify areas open to research, this review assesses the suitability of relevant techniques taken from each redundancy category to IFIS by:

- Comparing the characteristics of specific redundancy techniques and separating them into groups containing those suitable for computation and those suitable for communication.
- Assessing the communication techniques with respect to data rate, to the failure types expected in IFIS systems and to modular checker design.
- Assessing the computation techniques with respect to data rate, to failure type and to the generation of suitable communication codes.

## **2.3 HARDWARE REDUNDANCY**

All redundancy techniques aimed at on-line testing result in some hardware overhead with respect to their conventional counterparts. This section has however been limited to those techniques for which the hardware structure itself is the key to the provision of on-line testing.

### **2.3.1 Duplication with Comparison**

In duplication with comparison (DWC), reference data and its intended duplicate are calculated concurrently using independent hardware. While this technique provides error detection capabilities, the complexity penalty with respect to a conventional equivalent implementation is high [Al-Saad96]. However, duplication is recognised as being the most effective method for checking generic logical operations [Avizienis71a,Patel82]. This technique has been effectively incorporated into fault tolerant systems to increase system reliability by identifying faulty circuitry prior to its resultant removal from the system [Avizienis71a,Bartlett92].

### 2.3.2 Fault-Secure Networks

The output from a hardware duplication scheme can be thought of as a code. Fault-secure logic networks operate on and generate codes; however, the important feature of fault-secure networks is the structure of the network used to generate the code rather than the code itself. By definition, the sets of all codewords and non-codewords combined represent all possible input (or output) combinations for a given circuit. A circuit is said to be fault-secure if, for every fault from a prescribed set, the circuit never produces an incorrect code space output for code space inputs [Anderson73].

In [Ko78,Smith78] a technique is presented for designing fault-secure circuits. The technique involves identifying paths from internal fault sites that can affect more than one output under fault conditions, thus resulting in an incorrect codeword. Once found, hardware is added using minterm duplication and the paths are separated. Any fault is therefore guaranteed to affect only one output and consequently can only result in a non-codeword output when sensitised [Smith78]. Recently, techniques have been developed to synthesise fault-secure state-machines [Parekhji91,Bolchini95].

### 2.3.3 N-Modular Redundancy

N-modular redundancy (NMR) is a technique that is intended to provide fault masking in highly reliable systems. The underlying concept is to provide N means of calculating a function and propagate the most popular result. The unit responsible for selecting the most popular result is called a voter. A number of NMR configurations exist, where N is varied and/or the number of voting elements are adjusted resulting in a complexity/reliability trade-off [Russell89,Audet96]. For systems which are unsupervised for long periods, such as deep space missions, it was found that system reliability is maximised by maintaining a constant number of active elements which can take part in the voting process [Avizienis71a,Siewiorek92]. This is best realised by incorporating a number of additional, spare functional units that can be used as substitutes following identification and removal of faulty units [Russell89]. To adapt N-modular redundant designs to provide fault detection, in addition to masking, requires the addition of inequality detectors. A resulting disadvantage is that the voter(s) are never explicitly checked by the inequality checker.

### 2.3.4 Summary of Hardware Redundancy Techniques

Hardware redundancy techniques facilitate verification of data computation by employing circuit topology to ensure that normal calculations and reference information are not simultaneously subjected to the effects of single faults. All of the reviewed hardware redundant techniques exhibit the common feature that they do not affect the data rate during error-free operation. Consequently, these techniques do not incur error latency. Table 2-1 describes additional properties associated with hardware redundant techniques.

	Circuit Style	Error Coverage	Complexity ('C' means conventional)	Unchecked Hardware
<b>DWC</b>	Any	100%	$2C +$ Comparator	None
<b>Fault-secure Networks</b>	Combinational / Sequential	Code specific	$\leq (2C +$ Comparator)	None
<b>NMR</b>	Any	100% - Voter	$(N * C) +$ Voter + Comparator	Voter

*Table 2-1 Properties associated with hardware redundant techniques.*

The hardware redundancy techniques of DWC and NMR are not limited in the type of circuit to which they may be applied. Both can be applied to state machines, combinational designs and datapath structures. Conversely, fault secure networks do not appear to be suitable for protecting datapath structures. Hardware redundant design techniques exhibit no error latency and are applicable to all known error models, namely: symmetric errors, asymmetric errors and unidirectional errors.

In addition to those techniques specifically targeted at generating outputs for error detection, NMR techniques are intended to mask errors. The addition of an in-equality checker enables NMR designs to also be used for real-time error detection. However, this configuration does not perform any checks on the voting element itself, and consequently there is no guarantee that it is not calculating incorrect information.

While DWC offers the simplest concept targeted specifically at error detection, it is

that which results in the highest hardware complexity. Fault secure networks potentially offer the most efficient hardware implementation of the techniques presented, in the worst case being comparable to DWC. However, designing using the technique is more time consuming than with DWC because of the iteration necessary to ensure the independence of primary outputs under fault conditions.

Within the context of IFIS design, the above discussion suggests that DWC is the most suitable hardware redundancy technique.

## 2.4 INFORMATION REDUNDANCY

Information redundancy is the technique by which data is combined with additional information to form codes. The code words thus represent the information to be manipulated but are intended to be less vulnerable to errors. Codes can be error detecting and/or error correcting [Hamming50]. The context of this review is limited to on-line testing and therefore only error detecting codes are discussed. When selecting a coding strategy, the issues of error type, speed of operation, functional environment, code performance and the ease with which codes can be verified all require consideration [Pradhan80a]. The techniques for building error-detecting codes can be partitioned in the following way:

- Parity codes
- Arithmetic codes
- Unidirectional error-detecting codes

### 2.4.1 Parity Codes

These codes use checkbits that are calculated by performing modulo 2 addition on some or all information bits contained in the code. R.W. Hamming recognised the benefits of performing modulo 2 addition on transmission data to provide information redundancy capable of performing error detection [Hamming50]. The formation of parity codes dictates their error detecting properties and as such two distinct variations

exist, namely word-based and interlaced parity schemes.

#### 2.4.1.1 Word-based parity

Hamming defined parity as the modulo 2 addition of selected information [Hamming50]. To achieve error detection, a non-code word must be detectable under error conditions. In the case of a single error a non-code word must exist that differs in one bit position from the intended code word. Thus, successive Hamming codes must differ in at least two bit positions. This can be achieved by the addition of a single parity bit to existing data. Hamming showed that in general a code exhibiting a Hamming distance of 'd+1' can detect 'd' errors. [Siewiorek92] showed that by using a Hamming code with a distance of 2, any odd number of errors can be detected. Parity codes may be generated using independent calculation based upon circuit inputs (protecting computation) or by direct encoding of output data using modulo 2 arithmetic (protecting communication) [Fujiwara90,Goessel93].

#### 2.4.1.2 Interlaced parity codes

When modulo 2 addition is performed on a group of bits and an even number of errors occur within the group, these errors are not detected. However, by partitioning information into smaller groups and applying modulo 2 addition on each sub-group, it is only when an even number of errors occur within the same sub-group that errors become masked [Ko78,Fujiwara84]. The degree of error detection offered is dependent on the group size which in turn influences the hardware penalty incurred by the scheme. An extreme example of interlaced parity coding is demonstrated in the bitwise parity code (dual-rail code). In this case, 100% information redundancy is incurred by protecting each original data bit with its own parity bit [Lo93].

#### 2.4.2 Arithmetic Codes

Arithmetic codes are formed by applying some arithmetic function to the value represented by data bits resulting in a codeword. The original data may be directly retrievable (separable) from the resultant codeword or may require decoding (inseparable) [Siewiorek92, Piestrak95]. The separable codes considered are those

derived from the residue number system that was introduced in [Garner59]. The inseparable codes are those based on  $AN$  coding which was introduced in [Brown60].

#### 2.4.2.1 Separable arithmetic codes

Separable codes take the form  $N=\{X: X=DC\}$  where  $D$  is the data symbol and  $C$  is the check symbol. Using the residue numbering system developed in [Garner59], the check symbol (residue),  $C$ , for the data symbol,  $D$ , is  $|D|_b$ , where  $b$  is the base of the residue code. For residue codes it is therefore possible to predict the residue part of a code for the output of some arithmetic function when provided with only the residue of the inputs to that function. This means that residue codes can be used to check arithmetic operations in addition to communication (transfer) operations [Sayers86]. A variation of residue coding is inverse residue coding where each residue bit is inverted with respect to that of the equivalent residue code. Inverse residue codes are particularly well suited to transmission environments because they exhibit increased immunity to unidirectional errors [Avizienis71b]. In addition to the benefits offered by residue codes, a number of limitations exist, namely:

- Residue codes are not suited to checking division operations due to the difficulty in determining the absolute magnitude of a number associated with the residue [Garner59].
- To maintain a high error detection ability requires a large residue base [Avizienis71b]. Maximising the residue base maximises the number of available unique check symbols.
- The larger the residue base, the larger the number of bits required to represent it and therefore the more complex is the self-checking checker required to perform comparison [Russell89].
- While residue coding can be applied to logical operations, the efficiency with which the technique can be employed depends heavily on the exact function to be checked. In the case of an ALU the prediction of output residues often requires correction by mediation circuitry [Sayers85,Russell89].

### 2.4.2.2 Inseparable arithmetic codes

Arithmetic scaling of data can be used to calculate codes that are capable of error detection and correction [Brown60]. The codes, known as AN codes, are calculated using the form  $C = AN + B$ , where  $C$  is the codeword and  $A$  and  $B$  are constants with  $N$  being the original data. To allow negative numbers to be represented using AN error-detecting codes, the multiplier ( $A$ ) must be greater than 1 and odd [Brown60]. This provides codes separated by a minimum Hamming distance of 3.

It was noted in [Avizienis71b] that while the value of  $B$  was non-zero, specific knowledge of the arithmetic function being checked must be available. For example, for addition, the adder performs the function  $(A_i+B)+(A_j+B) = A(i+j) + 2B$ , where  $i$  and  $j$  are two operands. The validity of the result can be established by subtracting  $2B$  and showing that the result is divisible by  $A$ . For further operations to be performed on the result, its code should be modified by subtracting  $B$  to obtain  $A(i+j) + B$ . However,  $B$  need only be non-zero when optimising the code to exhibit specific Hamming distances in excess of 3. If error detection abilities for up to 2 simultaneous bit errors is sufficient, the hardware overhead required to realise AN coding is therefore minimal. Consequently, this class of AN codes are known as low cost arithmetic codes [Avizienis71b]. AN codes are not suitable for checking logical operations [Brown60, Avizienis71b, Russell89, Siewiorek92].

### 2.4.3 Unidirectional Error Detecting Codes

Unidirectional Error Detecting Codes (UEDs) are those codes that are designed specifically to detect unidirectional errors as described in section 2.2.1. UED codes can be sub-divided into three groups, namely:

- Unordered UED codes
- T-UED codes
- Burst-UED codes

### 2.4.3.1 Unordered UED codes

These codes exhibit the property that no codeword covers another in the same code set. Considering the relationship between two codes, X and Y: X is said to cover Y if X contains '1's in all positions in which Y contains '1's. Unordered codes can be subdivided further into systematic and non-systematic codes [Piestrak95].

Systematic unordered codes are formed by concatenating check bits and data to form a codeword. Systematic unordered codes are widely used for data transmission because they provide direct access to the message (data) content. This eliminates the need to translate received codes prior to message processing. A class of codes known as Berger codes has developed from an idea described in [Berger61] to exploit this characteristic. In Berger codes, which have been frequently reviewed, the checkbits are the inverse binary representation of the number of 1's in the data [Berger61, Frieman62, Ashjaee77, Pradhan80b, Smith84, Piestrak95]. This means that there must be enough checkbits to represent the maximum number of 1's in the data word. If D represents the number of data bits and C the number of checkbits, then the condition,  $2^C - 1 = D$  describes a maximum length Berger code. Not all transmitted data sequences require the full data space offered by Berger codes. In [Smith84] the properties of the data itself were exploited to minimise the number of unique check-symbols to be concatenated with the original data. This resulted in a more efficient coding but was specific for the data set employed. In [Ashjaee77] a technique for designing self-checking checkers (SCCs) for maximum length Berger codes was presented. Furthermore, an algorithm was developed which can process non-maximum length Berger codes to make them suitable for checking using an SCC.

Non-systematic unordered codes are typically weighted and in the case of m-out-of-n codes contain 'm' ones and 'n-m' zeros. They can detect all unidirectional errors (U-errors) because any U-error changes the weight (the number of 1's or 0's) of the code. The code capacity, the number of possible codewords, for any m-out-of-n code was proven to be maximal when  $m=n/2$  in [Frieman62]. Frieman also showed that for the same code capacity, the m-out-of-2m code is less redundant than Berger codes. M-out-of-n codes can be validated by using either a counter or a self-checking checker as described in [Anderson73, Piestrak95].



### 2.4.3.2 T-UED codes

In modern processing systems, it is common that data is processed as parallel words. It is reasonable to assume that once an error has occurred it will probably occur again because of repeated use of the same hardware. T-ued codes rely on the assumption that data transmissions are built from the concatenation of parallel data words. These codes are designed to detect up to 't' unidirectional errors contained within single words. They are therefore inherently less redundant than generic UEDs which are capable of detecting all unidirectional errors.

The number of concurrent unidirectional errors that can be detected using Berger (systematic) codes is determined by the number of checkbits [Dong86]. However, only errors in the information bits can be guaranteed detectable [Dong86]. Dong codes are modified Berger codes where the original checkbits are replaced by their dual-rail code equivalent. This makes the checkbits in Dong codes immune from unidirectional error masking. Bose-Lin codes offer an alternative method of protecting checkbits. In [Bose85] two different methods for checkbit partitioning were presented with the aim of choosing the most efficient checkbit generation algorithm that would result in unordered codes. The detection abilities of these codes depend only on the number of checkbits used, and not on the number of information bits. The first method uses a partitioning of '2:r-2' bits using a 1-out-of-2 code in the most significant 2 bits and the second partitioning scheme uses '4:r-4' bits with a 2-out-of-4 code. Figure 2-1 shows both code structures.

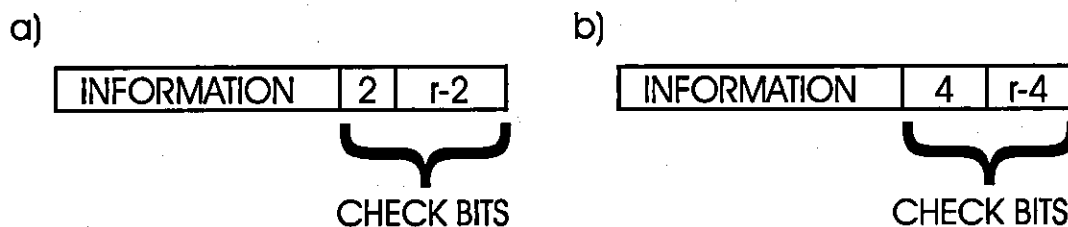


Figure 2-1 Bose-Lin codes: a) code type I, b) code type II

For values of  $r = 2, 3$  and  $6$ , code I is optimal because it can detect 2, 3 and 6 unidirectional errors respectively. For other values of  $r$ , code II is more effective. In

[Jha87] a systematic t-ued code was presented for which the capabilities depend on the relationship between the number of information bits and the number of checkbits.

Non-systematic t-UED codes were presented in [Borden82]. Borden codes are m-out-of-n codes that conform to specific criteria enabling them to detect 't' concurrent unidirectional errors. To exhibit this characteristic, they are designed so that  $m = \lfloor n/2 \rfloor \bmod(t+1)$ , where 'm' and 'n' are used in the conventional sense [Borden86].

### 2.4.3.3 Burst-UED codes

Efficient data coding can be achieved by combining multiple neighbouring data words into a block prior to coding [Grant89]. Burst-UED codes are capable of detecting bursts of errors spanning more than the length of a component data word.

Bose codes are systematic codes where the checkbits represent the weight of the information bits and are interleaved with them [Bose86]. The R checkbits are positioned so that some information bits are placed between the most significant checkbit and the rest. The code structure is shown in Figure 2-2.

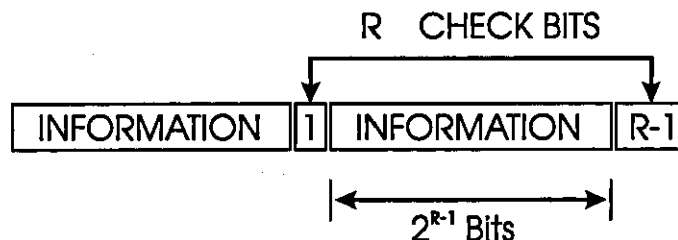
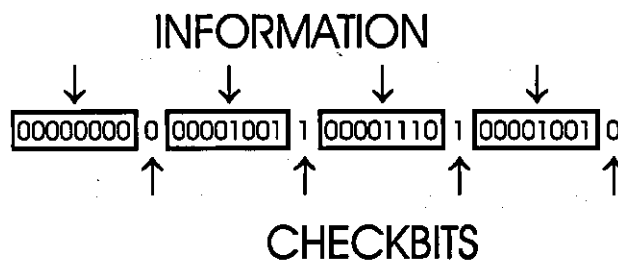


Figure 2-2 Bose code example.

A burst error capable of just spanning the length of the interleaved information cannot simultaneously cause an error in the checkbits. Thus, burst-ueds affecting the checkbits are detected in addition to those affecting data bits. The concept of exploiting interleaved checkbits was developed further by Blaum in [Blaum88].

Like Bose codes, the check bits in Blaum codes combine to form a symbol that represents the weight of the information bits. Unlike Bose codes, check-symbols are

coded so that the ability to detect unidirectional errors is maximised. The component bits of each check-symbol are interleaved with information bits to maintain the protective characteristics across the entire Blaum code.



*Figure 2-3 Blaum code example.*

Figure 2-3 shows the formation of an example Blaum code. In the example, suppose that information must be protected from a burst of up to 8 unidirectional errors [Blaum88]. This requires a minimum inter-symbol distance (between check-symbols) of 9 and can be achieved using 4 checkbits. To ensure that a burst of 8 or less unidirectional errors are detectable requires that 8 information bits are placed between each check symbol component. The Blaum code for the information burst '00000000 00001001 00001110 00001001' requires a check symbol that represents a data weight of 7 because there are 7 ones in the information. The specific symbol may be '0110' and is combined with the information bits in the way shown in Figure 2-3.

Blaum codes have been shown capable of detecting burst errors of up to  $2^{R-1}$  bits where  $R$  is the number of check bits in the code [Blaum88]. As  $R$  increases, this error detecting ability becomes asymptotic to  $2^R$ .

#### 2.4.4 Summary of Information Redundant Techniques

Like hardware redundancy, information redundancy does not impose error latency. Furthermore, information redundancy is attractive because a variety of coding techniques has been developed, each individually tailored to specific error models. This means that redundant information can be minimised to suite the operating environment.

Parity codes are simple and systematic, therefore incurring only a small complexity

penalty, being fast to calculate and not requiring codeword translation. By maximising message length, the efficiency of word based codes offering a specific degree of protection is increased. Conversely, interlaced parity schemes increase overall immunity to an even number of bit errors.

The check-base choice in arithmetic error detecting codes can significantly affect the complexity of the hardware implementation. Using either separable arithmetic codes or in-separable arithmetic codes, the best hardware efficiency is achieved when the total number of data symbols is exactly divisible by the number of check-symbols (modulus) [Avizienis71b]. In this case, where the codes are low cost codes, with check-base  $2^n-1$ ,  $n$  is called the group length. According to Wakerly, low cost AN codes and inverse residue codes with a group length of ' $n$ ' can detect all unidirectional errors of weight less than ' $n$ ' [Wakerly75]. Therefore, as the modulus (residue base) increases, the error detecting ability rises correspondingly. Finally, both separable and in-separable arithmetic coding methods are poorly suited to the checking of logical operations.

The algorithmic error detecting codes presented have all been specifically developed with the goal of detecting unidirectional errors. For the situation where bursts of data are transmitted, the direct application of non-systematic codes has not been deemed appropriate. However, non-systematic coding has been successfully and beneficially applied to the coding of checkbits which themselves are separable from the data bits contained in bursts.

Table 2-2 contains information extracted from the reviewed literature and summarises the properties of representative information redundant techniques. The suitability of information redundancy techniques to the communication and computation domains is contained within Table 2-2. The types of detectable errors are also included.

The number of concurrent detectable errors is dependent on the inter-codeword Hamming distance. For AN codes the Hamming distance between codes depends on the the modulus. Dong codes are capable of detecting  $<100\%$  of  $t$ -unidirectional errors [Dong84]. Dong codes are based on residue codes. As such, simultaneous errors are masked when their weight matches that of the residue modulus. As the

number of checkbits is increased, the number of possible occurrences is reduced for the same data width and thus the coverage becomes asymptotic to 100% [Dong84]. The abilities of Jha-Vora codes vary according to the ratio of checkbits to information bits,  $I$ . In some cases these codes were found superior to Bose-Lin codes [Jha87].

	Domain	Error Detection Type	Number of concurrent errors detectable using $R$ checkbits	Checkers TS(C),TS(T)
<b>Parity (even): Word</b>	Computation / Communication	Any single bit error	$R=1$ : Any odd number of errors	Hamming50(C), Russell89(C)
<b>Parity (even): Interlaced</b>	Computation / Communication	Any single bit error	$R=1$ per group Any odd number of errors in a group	Fujiwara84(T), Kakbaz84(T), Lo93(C)
<b>Residue</b>	Computation / Communication	Arithmetic, Logical	100% - all mod( $R$ ) valued deviations	Kundu96(C), Metra97(C)
<b>Inverse residue</b>	Computation / Communication	Arithmetic, Unidirectional	100% - all mod( $R$ ) deviations, 100% Unidirectional	Comparator: Lo93(C), Metra97(C)
<b>AN</b>	Computation	Arithmetic	(Hamming distance -1) implied by multiplier	No
<b>M-of-N</b>	Communication	Unidirectional	100%	Reynolds78(C), Piestrak96b(T)
<b>Borden (m-of-n)</b>	Communication	$t$ - Unidirectional, Symmetrical	100%, $m = \lfloor n/2 \rfloor \text{ mod}(e+1)$	Piestrak96a(T)
<b>Berger</b>	Communication	Unidirectional	100%	Ashjaee77(C), Chang96(C), Metra96(T)
<b>Bose-Lin (1)</b>	Communication	$t$ - Unidirectional	$R \in \{2,3\}$ : $R$ $R \leq 4$ : $2^{R-2} + R - 2$	Bose85(C)
<b>Bose-Lin (2)</b>	Communication	$t$ - Unidirectional	$R \geq 5$ : $5 \cdot 2^{R-4} + R - 4$	Bose85(C)
<b>Dong</b>	Communication	$t$ - Unidirectional	$< 100\%$	Dong84(C)
<b>Jha-Vora</b>	Communication	$t$ - Unidirectional	$R \geq 5$ : better than Bose-Lin (2) if $2^R \leq I < 1.42 \cdot 2^R$	Jha87(C)
<b>Bose</b>	Communication	Burst Unidirectional	$2^{R-1}$	Burns92(C), Bose86(C)
<b>Blaum</b>	Communication	Burst Unidirectional	$R \in \{2,3\}$ : $2^{R-1}$ $R \geq 4$ : $> 2^{R-1}$	Blaum88(C), Piestrak95(T)

Table 2-2 Properties of information redundant techniques.

The wide range of codes that have been developed necessitates the availability of a large number of different checker designs. If systems as a whole are to be trusted, the properties of the available checkers within them need to be established. Table 2-2 includes references to suitable checking techniques that have been developed for code checking. In the checker column, the checker techniques are provided along with the appropriate properties, namely: totally self checking (TSC) and totally self testing (TST) given by C and T respectively. Where neither of these properties exist, the checker does not check its own integrity.

## 2.5 TIME REDUNDANCY

The main disadvantage of hardware redundant schemes is the general overhead in circuit complexity. Similarly, a disadvantage associated with information redundant schemes is the need to encode/decode data thus potentially incurring a hardware penalty associated with calculation. In addition to this direct hardware penalty comes an increase in routing complexity and pin count needed for code transmission. Not all data processors require the continuous data throughput furnished by hardware redundancy and information redundancy [Patel82]. For such applications, where the metric of time is not fully exploited, the potential to provide reference data without the aforementioned drawbacks can be realised by using time redundancy. The time redundant techniques described are categorised within this review into three types namely those which incorporate functional re-mapping, those which incorporate functional partitioning and those based on duality.

### 2.5.1 Functional Re-mapping

Patel described a technique that allows efficient re-use of resources and yet provides a method for performing a calculation while furnishing an independent reference [Patel82, Patel83]. The technique was that of re-computing with shifted operands (RESO). During the first operation, the reference calculation is performed and the result stored. During the second operation, an equivalent calculation is performed using different paths through the same hardware structure as that used during the first operation. Figure 2-4 shows the concept of the approach.

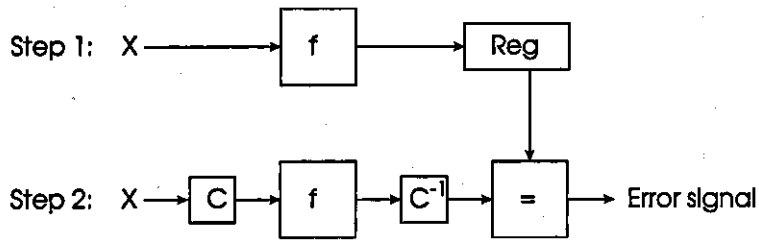


Figure 2-4 *Recomputing with shifted operands.*

In Figure 2-4, 'f' represents the hardware common to each processing step. X is the data input during both processing steps. The function labelled 'C' is chosen such that the output of 'C' causes different paths to be sensitised within f during step 2 to those that were sensitised during step 1. 'C<sup>-1</sup>' is some function that compensates for the effects of 'C'. In [Patel82], the function 'C' was a shift function which was applied to an arithmetic logic unit while in [Patel83] it was applied to multiply and divide arrays. Extensions to this technique were presented in [Laha83] to identify faulty partitions in a multiplier array prior to the masking of their effects.

In [Hana86] a variation of RESO was reported. The technique was that of recomputing with swapped operands (RESWO) and like RESO makes use of different signal paths to provide answers to be compared. Firstly, conventional functional calculation is performed on the input values. Re-computation is subsequently performed on reconfigured hardware. The reconfigured hardware is organised such that the section previously used to calculate the most significant part of the answer is now used to recalculate the least significant part, and vice-versa. An alternative variant on the theme of RESO was presented by Li and Swarzlander in [Li92] where shifting was replaced by barrel shifting thus requiring less hardware (RERO). To be sure that no interference occurs between the most and least significant bits of the pre-shifted operands, they are separated by the insertion of a zero. This requires an additional bitslice. These techniques are only applicable to iterative functions and are therefore not suited to Boolean operations.

### 2.5.2 Functional Partitioning

In [Johnson88] the application of a technique was described for performing a

calculation by combining the partial products resulting from two calculation steps (RESWO/DWC). In the example provided, a 32-bit adder, addition of the two least significant 16 bit words is performed twice in parallel during the first computation step. This allows immediate comparison of results. Following comparison, one of the results is stored in a register along with the overflow, and then the additions of the most significant 16 bit words are performed in parallel. The final answer is made up of the current adder outputs and the stored result. This technique while exhibiting time redundancy exhibits error latency typical of hardware or information redundancy due to the ability to perform checking following each processing step. However, this technique, like those based on RESO is only suitable for application to iterative designs because it relies on duplicate functionality between bitslices.

### 2.5.3 Duality

In [Shedletsky78] the technique of alternate data-retry was introduced as a method of fault masking. This technique requires that data be represented by codes. Furthermore, two codes, the bitwise complement of each other, exist for each data value. By checking the validity of received codes, a re-transmission can be requested using the alternate code for the last data sent. Assuming the single stuck-at fault model, the re-transmission is guaranteed to mask any single bit error. In [Reynolds78] a technique specifically targeted at error detection using self-duals was proposed. As in [Shedletsky78], functions are designed such that  $h(\bar{X}) \equiv \bar{h}(X)$ , if  $X$  is the input data set to the function. Computations are performed twice: once where the original data is supplied to the function inputs, and once where the bitwise complement is supplied. By combining the successive outputs, a dual-rail code is formed under fault-free conditions. It is important to ensure that the same computational path is not used during successive repetitive computations otherwise fault masking may occur [Shedletsky78, Reynolds78]. In the case of logical functions, this constraint may lead to a complexity increase in excess of 100%.

### 2.5.4 Summary of Time Redundancy Techniques

Time redundant techniques exploit the metric of time as an alternative mechanism to



hardware or information redundant techniques. The properties associated with time redundant techniques were obtained from the literature and are presented in Table 2-3.

	Design Style	Data Throughput (conventional = 100%)	Error Coverage	Error Latency	Complexity ('C' means conventional)
<b>Alternating Logic</b>	Any	50%	$\leq 100\%$	1	$\leq (2C + \text{Comparator} + \text{Double storage})$
<b>Alternate Data Retry</b>	Any	50-100%	-	1	$\leq 2C + \text{Comparator}$
<b>RESO</b>	Iterative Array	50%	100%	1	$(C + \text{Shift})/C + \text{Comparator}$
<b>RESWO</b>	Iterative Array	50%	100%	1	$C + \text{Comparator}$
<b>RERO</b>	Iterative Array	50%	100%	1	$(C+1)/C + \text{Comparator}$
<b>RESWO/DWC (REDWC)</b>	Iterative Array	50%	100%	$\leq 1$	$\cong 1.7 C$

*Table 2-3 Properties of time redundant techniques.*

With the exception of Alternating logic, the techniques specifically designed with the purpose of error detection all exhibit error coverage of 100% for single errors occurring in either the reference or calculation. In the case of Alternating Logic, the structure of the circuitry used to realise a self-dual function influences the maximum error coverage. The error coverage characteristics for Alternate Data-Retry were not encountered in the reviewed literature.

Time redundant techniques generally offer a potentially lower hardware penalty than hardware redundant techniques. Furthermore, recently developed time redundancy techniques remove the necessity to increase the data bus width between processing blocks with respect to their conventional counterparts. Therefore, for systems that contain blocks that do not need to compute during each clock cycle, time redundancy techniques may prove to be useful. For safety critical applications, most time redundancy techniques exhibit unacceptably high error latency.

## 2.6 ANALYSIS OF REVIEW

A restriction that influences the suitability of certain time redundant/hardware redundant techniques for adoption in IFIS is the lack of support for generic functions. For example, re-computation with swapped (or rotated) operands requires identical functionality between neighbouring bits within the datapath. This is not to be expected in the case of a decoder. Additionally, techniques of hardware redundancy and time redundancy have been found only to provide information that supports dynamic testing of computation, while information redundancy provides techniques for both computation and communication.

This information has resulted in the ability to exclude techniques that are not suitable for adoption within IFIS. Further conclusions can be drawn by closer examination of the remaining reviewed techniques with respect to the characteristics directly related to communication and computation.

### 2.6.1 Coding between Computation Elements

Table 2-4 shows the information redundant techniques that are applicable to data communication. An important consideration when choosing a coding technique is the ease with which a code can be built and checked in addition to the rate at which the data can be extracted for subsequent calculation. This is related to the hardware required to calculate and check the code and the ability to access the data contained in the code. Clearly, codes requiring ROMs and counters (sequential logic) are more difficult to calculate and check than those built using combinational techniques. Similarly, separable codes permit concurrent data access and code checking.

The failure types for which IFIS is intended were identified as unidirectional errors and those resulting from stuck-open/bridging (combinational/sequential) faults. IFIS was additionally specified to use encoding that supports modular checker design.

Based on the above discussion, only bitwise even parity coding or 1-out-of-2 coding can be built/checked fast while also supporting modular checker design. Both result in 100% information redundancy forming a dual-rail code.

	<b>Encoding Method / Type</b>	<b>Failure Type</b>	<b>Supports Modular Checker Design</b>
<b>Parity (even): Word</b>	Xor/Separable	Any single bit error	No
<b>Parity (even): Interlaced</b>	Xor/Separable	Any single bit error	Bitwise
<b>Residue</b>	Coder/Separable	Arithmetic	No
<b>Inverse Residue</b>	Coder/Separable	Arithmetic, Unidirectional	No
<b>M-of-N</b>	ROM / Inseparable	Unidirectional	1-of-2
<b>Borden (m-of-n)</b>	ROM / Inseparable	t-Unidirectional, Symmetrical	No
<b>Berger</b>	Counter / Separable	Unidirectional	No
<b>Bose-Lin (1)</b>	Counter or adder tree / Separable	t-Unidirectional	No
<b>Bose-Lin (2)</b>	Counter or adder tree / Separable	t-Unidirectional	No
<b>Dong</b>	Separable	t-Unidirectional	No
<b>Jha-Vora</b>	Counter + ROM / Separable	t-Unidirectional	No
<b>Bose</b>	Counter or adder tree / Separable	Burst Unidirectional	No
<b>Blaum</b>	Counter + ROM / Separable	Burst Unidirectional	No

*Table 2-4 Properties of codes suitable for communication*

Bitwise odd parity dual-rail codes detect unidirectional errors because unidirectional errors result in even parity (non-codewords). However, the even parity scheme cannot detect unidirectional errors because they result in incorrect codewords.

Data coding alone is insufficient to guarantee the detection of errors resulting from stuck-open or bridging faults because they typically affect the time taken for a correct value to occur. To detect them requires checking for corrupt data at specific points in time. To ensure that the codes currently appearing at checker inputs are up-to-date requires the ability to differentiate between successive sets of codewords.

An example of such a scheme is where consecutive dual-rail codewords exhibit alternating parity, thus ensuring monotonic transitions. Monotonic transition

checking, in the form of dual-rail return-to-zero coding, was proposed to provide on-line testing in [Jones91]. However, the scheme was not characterised in this context. Additionally, saturated (four-phase) dual-rail coding schemes (where all codes carry data) have been employed to co-ordinate computation within asynchronous designs by using monotonic code transitions to signal completion detection [Linder96,David92,Dean91]. The advantages of applying such coding schemes to on-line testing are that:

- Successive sets of codewords can be distinguished by alternating parity.
- Single errors are detected by the appropriate parity code-set. These may result from combinational (temporary) faults or from sequential faults.
- Unidirectional errors (more than one bit) lasting more than one processing cycle enforce fixed (even) parity, thus being detected by the inability to assume odd parity.

### 2.6.2 Design Techniques for Computation

IFIS computation elements accept and compute communication codes. The data content of input codes is operated upon to generate the data content of output codes. The IFIS communication codes can be computed in one of the following ways:

- As a direct result of data computation.
- Following data computation and based on its validity.

If communication codes are to be generated as a direct result of data computation the computation circuit should generate outputs that are similar to those required to protect communication. If the outputs generated directly from data computation are not similar to those required, then a complex mapping function may be required. Alternatively, communication code generation may be performed based on the validity of data computation. Valid IFIS codes are assigned to IFIS cell outputs if data computation is verified. Otherwise, invalid IFIS codes are assigned. Table 2-5 shows the on-line test techniques from those reviewed that are specifically targeted at facilitating error detection within the computation domain.

	Data Rate	Failure Type	Dual-rail Compatibility
<b>DWC</b>	100%	All logical errors	Inherent (duplication)
<b>Residue</b>	100%	Arithmetic	Not Inherent
<b>Inverse Residue</b>	100%	Arithmetic / Unidirectional	Not Inherent
<b>Alternating Logic</b>	50%	All logical errors	Inherent (duplication)

*Table 2-5 Properties of computation techniques that can be applied to any digital function.*

Duplication with comparison (DWC) and alternating logic both compare duplicate data thus, they can reasonably be expected to be compatible to dual-rail coding. However, residue codes rely on a representation of data that requires translation before direct comparisons can be made. These techniques are therefore not expected to be directly compatible to dual-rail code generation.

### **2.6.3 Areas Open to Research within the Context of IFIS.**

IFIS designs exhibit strong similarities to those asynchronous systems that use dual-rail codes. However, saturated dual-rail coding schemes do not appear to have been applied to on-line testing. Furthermore, the properties associated with the dual-rail return-to-zero code proposed in the original IFIS patent were not characterised [Jones91]. The lack of available information regarding such applications suggests that the application of dual-rail code sequences to on-line testing warrants attention.

Having identified dual-rail coding coupled to a sequential protocol as candidates worthy of investigation, circuit structures that support their implementation should also be investigated. In section 2.6.2, it was highlighted that no quantitative information is available regarding the suitability of specific redundancy techniques to support the generation of dual-rail code sequences. To quantify the characteristics of the selected redundancy techniques to dual-rail code generation, requires further investigation.

By investigating both of the specified areas, the direct and indirect influences of coding and protocol on IFIS system characteristics can be quantified.

# **CHAPTER THREE**

## **OVERVIEW OF INVESTIGATIONS**

### **3.1 OBJECTIVES OF CHAPTER**

The objectives of this chapter are to select and introduce the investigations contained within this thesis. More specifically, the objectives are to:

- Provide a statement of research objectives
- Present the status of the IFIS design methodology prior to this research
- Introduce the proposed investigations

### **3.2 OBJECTIVES OF RESEARCH**

This thesis focuses on the IFIS on-line test methodology. More specifically, this thesis provides an assessment of different aspects of the methodology and offers evidence as to its suitability for application to on-line testable systems.

#### **3.2.1 Identification of Research Topics**

Chapter 1 identified the possibility to perform failure diagnosis in IFIS systems based on the halting sequence of IFIS system outputs.

Chapter 2 highlighted the need to investigate the implications of using dual-rail code sequences to provide on-line testing. The specific coding schemes that were identified were saturated-four-phase coding and return-to-zero coding. These schemes exhibit the common feature that consecutive codewords are distinguishable. To support this feature, transitions must conform to a suitable protocol.

### 3.2.2 Statement of Research Objectives

This thesis aims to understand the impact of the IFIS on-line test methodology on system design. The following objectives have been identified to enable an assessment to be achieved:

- To quantify the restrictions on performance and test coverage imposed by different dual-rail coding and protocol schemes and select a suitable combination for IFIS.
- To assess the characteristics of test coverage, complexity and performance associated with different circuit design techniques used to implement the chosen IFIS coding scheme and protocol.
- To characterise a demonstration IFIS system that incorporates multiple computation elements that are interconnected such that the IFIS protocol is enforced.
- To quantify the diagnostic resolution attainable based only on the halting sequence of primary system outputs following error detection.

The issues relating to test coverage, complexity and performance are all issues that directly influence the potential future use of the IFIS methodology.

Failure diagnosis is important because it can be used to assist in the repair of repairable systems, thus reducing or eliminating system outage. Furthermore, it can be used to identify weak-points in design, thus permitting preventive measures to be used in future implementations. If the inherent diagnostic ability of IFIS systems is sufficient to facilitate system repair, then IFIS designs may increase system availability because diagnosis occurs during system shutdown rather than afterwards.

### 3.3 STATUS OF THE IFIS METHODOLOGY

IFIS (If it Fails It Stops) is currently under investigation at Loughborough University

and is the subject of the investigations contained in this thesis. The status of IFIS at the beginning of this research is summarised below.

A patent was issued in 1991 describing the concept of applying asynchronous communication techniques in the on-line testing environment [Jones91]. This patent specified that coding be employed to carry data and cell status information.

System level modelling was needed to explore the data throughput and halting properties associated with combinations of different dual-rail coding schemes and inelastic/elastic protocols. This motivated a group effort to generate behavioural VHDL models for IFIS processing elements. At that time, the potential implications of protocol, encoding and processing element structure on testability, performance or halting behaviour had not been quantified. Furthermore, the identification of the potential to use halting sequence as a means of failure diagnosis had not occurred.

### **3.4 INTRODUCTION TO INVESTIGATIONS**

This section introduces each investigation, detailing the issues to be addressed and the approaches applied to answer them.

#### **3.4.1 Selecting a Version of IFIS: The Influences of Data Coding and Protocol**

The behavioural specification of the IFIS design methodology is influenced by the halting mechanism itself. This influence stems from the requirement to ensure error containment: a task passed to external circuitry following the recognition of an error by detection circuitry in conventional systems. The objectives of this investigation are to:

- Introduce the requirements of designs implemented using an on-line test methodology which uses halting as a mechanism for error flagging and data management.



- Understand the potential influences of data coding and protocol on the testability, behaviour and performance of IFIS designs.
- Select the combination of data coding and protocol that exhibits the most attractive attributes.

This investigation examines the properties associated with different coding techniques and protocols. IFIS has already been defined to use dual-rail coding where two rails are used to encode each information bit. The exact dual-rail code employed can affect the following:

- Data throughput rate
- The code space

The data rate influences the attractiveness of the design methodology when considering it as a candidate for adoption in future designs. The code space restricts the range of vectors that can be applied to processing elements and thus may limit the ability to stimulate fault sites within them.

The protocol controls the activity of processing elements. The choice of protocol to be adopted can affect:

- The data throughput rate
- The code space
- The behaviour

The behaviour of IFIS systems under normal operating conditions depends on the type of protocol used, while the behaviour following fault-conditions also depends on the selection of protocol monitoring points.

This investigation examines dual-rail code and protocol combinations in the above context and identifies a suitable combination to apply to future IFIS designs.

### 3.4.2 IFIS Cell Design

The structure of designs is known to affect the properties associated with them. Within the context of design-for-test, circuit structure is tailored to provide the most acceptable combination of test coverage and hardware penalty for the specific application. The objectives of this investigation are to:

- Understand the influence of applying different design techniques to the generation of reference information and verification/error management circuitry within IFIS processing elements.
- Select that structure which provides the best features in the context of IFIS on-line testable systems. The selected structure will then be adopted as a template for future designs.

This investigation examines the influence of the circuit structure applied to IFIS processing elements on the following:

- Testability
- Data throughput
- Complexity

These attributes were chosen because they are those which are commonly used to distinguish between design-for-test techniques. Cell partitioning is considered and a number of alternative structures that were described in the review are applied to processing element sub-blocks. The options, together with alternative proposals are quantitatively compared and a generic IFIS processing element architecture is developed.

### 3.4.3 IFIS Feasibility Study

The previous two investigations independently addressed issues related to the IFIS processing elements and the connections/communications between them. The

feasibility study aims to:

- Report the application of IFIS to a real-world design implemented in hardware.
- Outline the design experiences associated with designing using IFIS.
- Demonstrate the operation of the circuit both in fault-free mode and with a specific failure.
- Compare the performance, complexity and fault coverage of the conventional and IFIS equivalent implementations. This empirically quantifies the characteristics of a combined system built upon the results of the previous two investigations.

The feasibility study exploits the results obtained from the previous two investigations and applies them to the re-design of a commercial UART. The UART was considered to be a representative design because it includes busses, state machines and registers, all of which are frequently used design elements.

To assess the effectiveness of the methodology a comparison is made between a conventional UART and its IFIS equivalent implementation on FPGA. The UART represents one of the most complex designs using a single on-line test technique to date. Output traces are shown for the IFIS implementation on FPGA operating under fault-free conditions and with deliberate failures injected. Comparisons of size and speed are presented in addition to an indication of on-line test coverage.

#### **3.4.4 Failure Diagnosis in IFIS**

This investigation presents and assesses a failure diagnosis technique that exploits the halting properties associated with IFIS designs. The aims of this investigation are to:

- Develop a strategy that is appropriate for performing failure diagnosis in designs that have been implemented using the IFIS on-line test methodology.
- Describe the fault models that are to be applied to IFIS designs.

- Identify a relationship between the behaviour of models to simplify algorithmic diagnosis.
- Show that the technique is suitable for automation by presenting algorithmic approaches to failure diagnosis in IFIS.
- Compare and select the most efficient algorithm for future use.
- Present the failure-diagnosis view of the experimental vehicle.
- Perform failure diagnosis on the experimental vehicle under a variety of fault conditions.
- Assess the diagnosis technique within the context of the investigation.

With the aim of developing an efficient method of failure diagnosis for use with IFIS designs, the essential capabilities required to perform failure diagnosis are identified. The facilities inherent in IFIS designs are identified and described within the context of failure diagnosis.

Having established which properties are provided by IFIS it remains to develop a complementary technique that provides the missing attributes required to perform fault diagnosis. A technique for fault diagnosis is proposed that exploits the halting properties associated with IFIS designs.

Based on the logical feedback structure of IFIS cells, logical fault sites with differing halting effects are identified with respect to the cell structure to form an expanded fault list. Algorithms are presented which can generate dictionaries for different logical fault sites (fault classes). The performance of these algorithms is compared and one is selected for application to the UART used in the feasibility study under different injected fault conditions.

The individual faults that can be injected in the IFIS UART and the effects of combining them to emulate other fault conditions are described. Failure diagnosis is performed on the IFIS UART containing a number of different injected fault conditions. The ability to distinguish between single occurrences of the extended fault list is quantified empirically.

## CHAPTER FOUR

### SELECTING A VERSION OF IFIS

#### 4.1 OBJECTIVES OF CHAPTER

The behavioural specification of the IFIS design methodology is influenced by the halting mechanism itself. This influence stems from the requirement to ensure error containment: a task passed to external circuitry following the recognition of an error by detection circuitry in conventional systems. The objectives of this investigation are to:

- Introduce the requirements of designs implemented using an on-line test methodology which uses halting as a mechanism for error flagging and data management.
- Understand the potential influences of data coding and protocol on the testability, behaviour and performance of IFIS designs.
- Select the combination of data coding and protocol which exhibits the most attractive attributes.

#### 4.2 INTRODUCTION

IFIS (If it Fails It Stops) is an on-line test methodology that uses halting as a mechanism to provide on-line test features [Jones91]. The halting mechanism is provided by a protocol that permits/prohibits data flow according to the validity of received data. The data is coded to provide the necessary information for validation according to the protocol. It was suggested that dual-rail coding be the technique applied to IFIS in [Jones91]. To assess the implications of employing different coding

and protocol techniques, the expectations of IFIS in the context of on-line testing must be defined. For any on-line test methodology, the following attributes are important:

- Behaviour under fault-free and faulty conditions
- The on-line test coverage
- The performance and complexity

The behaviour of IFIS designs must be such that the inherent error management only permits correct (uncorrupted) data to propagate. The implications of this statement are that IFIS must detect incorrect data, prevent the incorrect data from propagating and prevent future data propagation. The permanency of halting is necessary because it is not only the validity of data at any single point in time that is important, but also the validity of the complete data burst. If corrupted data were removed from a data burst, the received data burst would be different in length from the transmitted burst and must therefore be corrupt.

In addition to the behaviour of designs under faulty and fault-free conditions, it is important that the IFIS on-line test methodology be capable of checking a high proportion of the design during normal operation. In the context of off-line testing, the measure of quality applied to a test vector set is the test coverage. The test coverage of a vector set refers to the percentage of modelled faults that are detectable for a specific design when the vector set is applied. Similarly, on-line test coverage is intended to be a measure of the ability to detect modelled faults. Unlike the off-line test environment, the test vector set is neither constrained in size nor data content. However, the data in IFIS is coded to permit protocol checking. The data coding therefore restricts the range of vectors that can be applied to IFIS processing elements and may limit the maximum on-line test coverage.

As with any other design methodology, maximal data throughput, minimal complexity overhead and maximal ease of implementation are all factors that affect the attraction of the methodology to potential users.

This investigation examines dual-rail code and protocol combinations in the above context and identifies a suitable combination to apply to future IFIS designs.

### 4.3 CODING

When transmitting messages from one location to another, the message is built from a sequence of symbols (or codewords). These symbols are decoded upon message reception to extract the original data content. Multi-rail encoding schemes supplement data with additional information that can be used to provide completion detection (to distinguish between symbols) in an asynchronous environment or endow error checking in a test environment. The haltability and error propagating characteristics of asynchronous circuits were reported by Varshavsky in [Varshavsky86].

Dual-rail coding is one implementation of the general set of multi-rail encoding schemes and is well established. IFIS has already been defined to use dual-rail coding, where two rails are used to encode each information bit, to carry data and status information between cells. The exact dual-rail code employed can affect the following:

- Data throughput rate
- The code space

The data rate influences the attractiveness of the design methodology when considering it as a candidate for adoption in future designs. The code space restricts the range of vectors that can be applied to processing elements and thus may limit the ability to stimulate fault sites within them. Two dual-rail coding schemes which are commonly applied to asynchronous systems were selected for examination namely, return-to-zero coding and saturated coding.

### 4.3.1 Return to Zero Coding

Return-to-zero (RTZ) coding is widely used in telecommunications systems. Within the domain of integrated circuit engineering, Mead and Conway introduced it as a means of delivering sequence information within asynchronous circuits [Mead80].

Figure 4-1 shows the state transition diagrams for the conventional binary coding scheme and the RTZ coding scheme. The binary coding scheme is not suitable for providing completion detection or error detection because it does not possess sufficient code space to permit representation of data and supplementary information. In binary systems, there are no constraints on state transitions, i.e. a '0' can be followed by a '0'. To ensure that codes representing data and error status within IFIS designs are current, successive code states must be distinguishable.

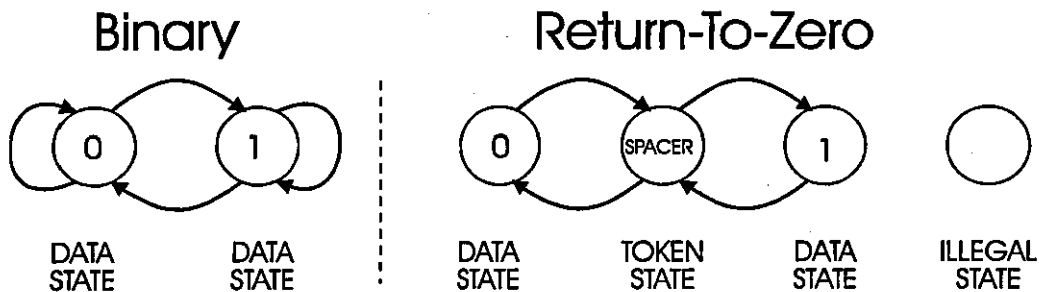


Figure 4-1 State transition diagrams for Binary and Return-to-zero systems.

In the RTZ coding scheme shown in Figure 4-1 data is supplemented with additional information by increasing the code space. The code space is increased to include a spacer that enables transitions to be observed between data states of identical value. Only those transitions shown in the (RTZ) state diagram are considered error-free. Figure 4-1 shows that self-transitions are prohibited when using such schemes.

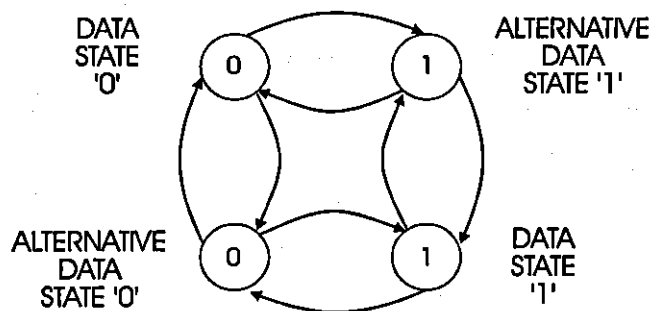
The maximum data transfer rate associated with RTZ coding is restricted because not all symbols carry useful data. When implemented using binary logic levels, the RTZ scheme requires two rails to encode the three states, namely: data state 0, data state 1 and the spacer. This results in a spare (illegal) state.



### 4.3.2 Saturated Four Phase Coding

Saturated-four-phase dual-rail coding was used to provide completion detection in asynchronous designs in [Dean91, David92, Linder96].

Figure 4-2 shows the state transition diagram for a saturated dual-rail coding scheme. As with the RTZ coding scheme described in section 4.3.1, self-transitions are not permitted. Comparison with Figure 4-1 reveals that the illegal state and spacer associated with the RTZ scheme have been replaced by alternative data states.



*Figure 4-2 State transition diagram for saturated coding scheme.*

This saturated coding scheme maximises the capacity of the code because each state is a data carrying state. Thus the data rate is the same as the symbol rate. Figure 4-2 shows that two codes represent each data value (0 and 1) and yet transitions between states are still constrained. This type of saturated coding scheme was employed to encode data in an asynchronous environment in [Dean91].

### 4.3.3 Application of Codes to IFIS

As specified in section 4.3.1 transitions between successive codes must be verifiable. The following options for verification have been identified:

- Translate the codes appearing at the processing elements into state labels and verify the state sequence.

- Choose codes such that an easily identifiable relationship between successive codes exists.

Code translation could be performed and valid state neighbours identified from the previous state using a dictionary. While this is an option, it is complex and therefore not attractive for such a fundamental process.

Identification of a relationship between successive codes revealed that the same dual-rail code representation used to provide completion detection in asynchronous designs could be adopted for error detection. The particular dual-rail coding scheme ensures that only monotonic transitions occur when adhering to a valid state sequence. Two advantages of this option are:

- The code used exhibits closure and is therefore suitable for the saturated scheme in addition to the RTZ scheme.
- Transitions can be easily checked based on the alternating parity of successive codes.

Table 4-1 relates the dual-rail code representations and meanings in the context of the RTZ coding scheme and in the context of the saturated coding scheme. The dual-rail code is carried on a pair of single-bit signals, nominally labelled *t* and *f*. An asterisk denotes the alternative data states, as shown in Figure 4-2.

Bit 't'	Bit 'f'	RTZ Code Name	Saturated Code Name
0	1	0	0*
0	0	Spacer	0
1	0	1	1*
1	1	Illegal	1

*Table 4-1 Dual-rail codes and their meanings.*

In Table 4-1 the rows have been arranged such that neighbouring rows represent valid neighbours in both the RTZ and saturated coding schemes described in section 4.3.1 and section 4.3.2.

Considering that the codes are carried by signal lines, the effect of a logic-affecting fault occurring on one of those signal lines is to change the possible transitions between states. For example, considering the saturated coding shown in Table 4-1 in conjunction with Figure 4-2, a stuck-at-1 on the  $t$  line will make it impossible to represent 0 or  $0^*$ . Thus the arcs from state  $1^*$  to state 0 and from state 1 to  $0^*$ , shown in Figure 4-2, are removed. Hence if the last state received by a circuit element was 1 ( $1^*$ ) and an upstream element is attempting to write a  $0^*$  (0) on the faulty lines, that transition will never complete. As a result, the receiving element therefore does not process.

#### 4.4 PROTOCOL

Received data codes are verified for adherence to the protocol that in turn controls the activity of processing elements. The choice of protocol to be adopted can affect:

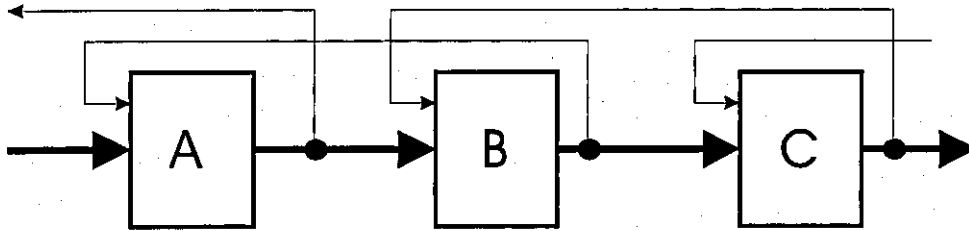
- The data throughput rate
- The code space
- The behaviour

The behaviour of IFIS systems under normal operating conditions depends on the type of protocol used, while the behaviour following fault-conditions also depends on the selection of protocol monitoring points. Protocols can be partitioned into elastic protocols and inelastic protocols. The fundamental concepts of these categories are described.

##### 4.4.1 Elastic Pipelines

Elastic pipelines are those formed by the interconnection of asynchronous processing elements as described in [Sutherland89]. Figure 4-3 shows the interconnection of processing elements to form a generic pipeline structure combining actual data with data control signals, as is the case where dual-rail encoding is employed. The heavy arrows, from left to right, represent data flow, while the lighter arrows, right to left,

represent feedback. Two phase handshaking can be performed using a dual-rail encoding scheme by treating data-flow states as a source of requests and feedbacks as a source of acknowledgements.



*Figure 4-3 Interconnection of processing elements 'A', 'B' and 'C' forming a pipeline.*

An elastic pipeline is a variable depth symbol buffer. The pipeline can contain a variable number of symbols between the minimum symbol buffer depth and the maximum symbol buffer depth, the latter state being that of saturation.

The number of symbols held in the pipeline is dependent on the rates of symbol supply and removal. If the removal rate is lower than the supply rate, eventually the pipeline becomes saturated thus limiting the supply rate to be the same as the removal rate.

If the removal rate is faster than the supply rate, the contents of the pipeline eventually reach a minimum. In this situation, all elements contain a function of a copy of the last symbol supplied to the pipeline. For example, if each of the elements shown in Figure 4-3 act only as temporary storage, then all elements in the pipeline contain a copy of the last symbol supplied to element A. As soon as a new symbol is supplied, it progresses through the complete pipeline and appears at C, leaving each element in the pipeline containing a copy of the symbol.

Figure 4-4 shows the sequence of events that might occur in an elastic pipeline. For the sake of generality, two symbol types are shown. **Symbol type 1** represents the data state 0 or 1. **Symbol type 2** represents the token state (spacer) for the RTZ coding scheme and represents the alternative data state for 1 or 0 for the saturated coding

scheme. As was the case in Figure 4-3, conventional data (symbol) flow is from left to right, while feedback, described above, flows from right to left.

In Figure 4-4, i) shows a stable saturated state; meaning that none of the processing elements A, B or C can process symbols. According to the elastic protocol, the pipeline cannot accept new symbols until a symbol is removed downstream. Therefore, this state remains until a change on the feedback input to cell C occurs as a result of downstream symbol removal. ii) As soon as data changes on the feedback supplying element C, the protocol condition for the element is satisfied, thus allowing a new element output to occur in step iii). Step iv) shows that element B has now processed as a result of step iii), thus allowing element A to process during the next clock cycle. This results in the state shown in step v).

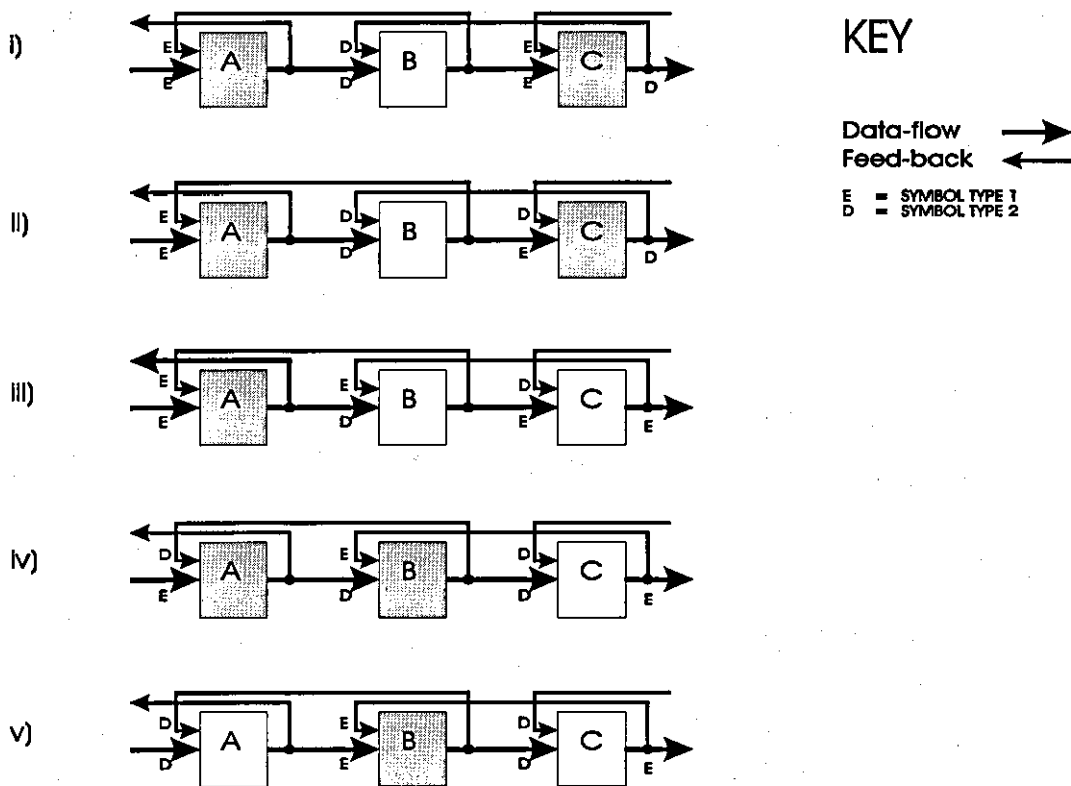


Figure 4-4 Data movement in an elastic pipeline. i) the stable state, ii) a change on the feedback input to cell C at time 0. iii) new output of cell C one clock cycle later at time 1. iv) at time 2 the output of cell B has changed. v) the next stable state.

Examination of Figure 4-4 as a whole, shows that symbols are copied (processed) by the immediate neighbouring downstream element before being overwritten by an

upstream neighbour. For example, the symbol held in element B is copied to element C during step iii prior to being overwritten by a copy of the symbol held in element A during step iv. This is characteristic of elastic pipelines.

It may be noticed that during step iii it would have been possible to change the data applied to the feedback to element C. This is what would happen if data was continually removed from the pipeline. By repeating steps iv and v while also updating the feedback input to element C once every two clock cycles, the maximum symbol rate of one element per two processing cycles is achieved.

#### 4.4.2 Inelastic Pipelines

Inelastic pipelines contain a constant number of symbols. This is because symbols move in lock-step throughout the pipeline. As a symbol appears at the output of the pipeline, so a new symbol must appear at the input to the pipeline. Symbols are not duplicated within inelastic pipelines.

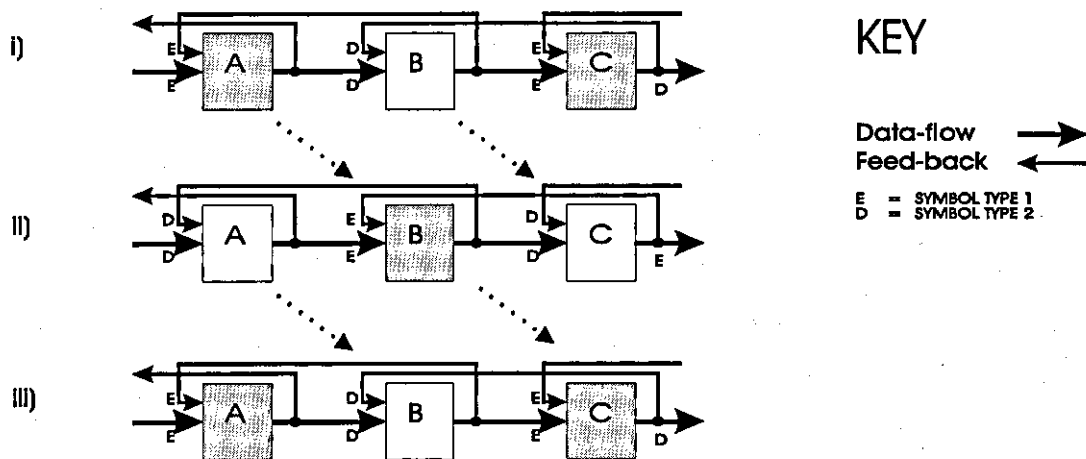


Figure 4-5 Data movement in an inelastic pipeline. i) the pipeline at any nominal time  $t$ .  
ii) the state of the pipeline at  $t+1$ . iii) the state of the pipeline at  $t+2$ .

The inelastic protocol requires that the quantity of data held in the pipeline remains constant. For this reason, data is supplied to the pipeline at one end while data is simultaneously removed from the other. As a result, the data held within the pipeline progresses from left to right at the rate of one cell per clock cycle.

### 4.4.3 Application of Protocols to IFIS

Figure 4-3 shows a generic pipeline that allows bi-directional transfer of information between neighbouring processing elements. In the asynchronous environment, the information is used to control data flow based on the readiness of downstream processing elements to receive new data. However, in the IFIS environment, the information is used to control data flow based on its validity as interpreted by the current processing element.

Prior to code validation in IFIS, time must be allowed for codes to stabilise. This necessity is eliminated in the asynchronous environment because precautions are taken to avoid race hazards, thus eliminating intermediate values from occurring during transitions. The precautions taken in asynchronous designs are often costly in terms of complexity and should be avoided if possible.

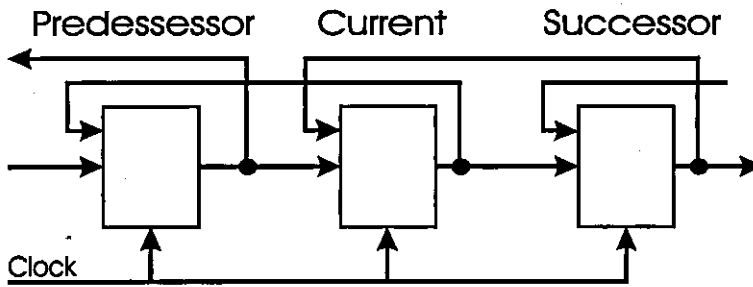
The need to avoid the occurrence of intermediate code transitions in IFIS designs is eliminated by ensuring that validation only occurs following a predetermined passing of time. IFIS designs are synchronous and therefore all internal states are expected to have stabilised prior to consecutive active clock edges. Validation is therefore applied only to codes that have been stored in synchronous registers.

### 4.5 BEHAVIOUR OF ALTERNATIVE IFIS VERSIONS

The dual-rail protocols used in asynchronous designs are designed to allow processing elements to resume processing as soon as specific requirements have been fulfilled. The selection of nodes that are monitored to provide handshaking in IFIS can influence the behaviour following the occurrence of a fault condition. The nodes conventionally monitored in asynchronous designs are used as a basis for the protocol in IFIS designs and the resultant halting behaviour is compared against the requirement described in section 4.2. A new protocol is proposed and its behavioural attributes are verified.

### 4.5.1 Existing Protocol Conditions

Figure 4-6 shows three neighbouring cells within an IFIS pipeline. As was the case in Figure 4-3, data flow is from left to right while feedback is from right to left. All cell inputs appear on the left hand side of each cell while cell outputs appear on the right. The current cell refers to any cell that is currently under examination within the pipeline. In Figure 4-6 it is apparent that the **current** cell monitors the **predecessor** and **successor** cells by means of their data and feedback connections respectively.



*Figure 4-6 Neighbouring elements within an IFIS pipeline.*

The activation of the current cell depends on the compliance of the monitored outputs to the protocol.

#### 4.5.1.1 General behaviour

The behaviour of the current cell can be described in the following way:

Iff (the protocol condition is satisfied)  
 then  
     the next cell output =  $f(\text{current input})$   
 else  
     the next cell output = the current cell output

The protocol condition depends on the coding scheme and the type of pipeline employed. According to the discussion in section 4.3.3 the codes presented in Table 4-



1 can be classified into code sets such that codes of equal parity are placed in the same set. For clarity, this classification is presented in Table 4-2.

Generic Label	RTZ Coding Scheme	Saturated Coding Scheme
Set 0	{0,1}	{0,1}
Set 1	{spacer}	{0*,1*}

Table 4-2 Code classification.

Considering that the union of Set 0 and Set 1 within a specific coding scheme represents all codewords in that scheme, then let the union of Set  $x$  and Set  $\bar{x}$  represent all codewords in that scheme.

If the outputs of the predecessor, current and successor cells are labelled as  $S(p)$ ,  $S(c)$  and  $S(s)$ , respectively then the current cell behaviour for a cell according to an elastic protocol is given below.

```

Iff ( (S(p) = Set x) and (S(s) = Set  $\bar{x}$ ) )
    then
        next S(c) = f( S(p) )
    else
        next S(c) = S(c)

```

Similarly, for the inelastic protocol, behaviour is described in the following way.

```

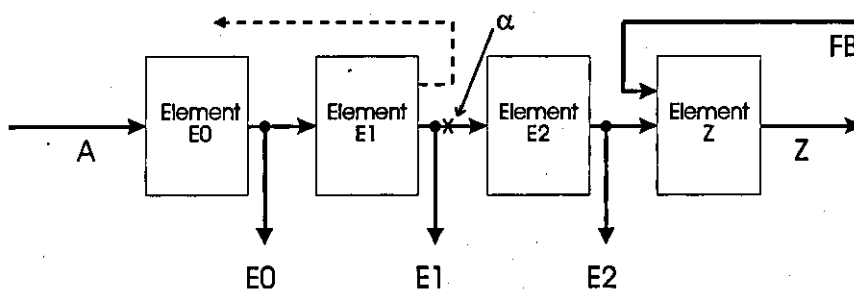
Iff ( (S(p) = Set x) and (S(s) = Set x) )
    then
        next S(c) = f( S(p) )
    else
        next S(c) = S(c)

```

### 4.5.1.2 Faulty conditions

The behaviour exhibited by the both the elastic and inelastic protocols under normal conditions was presented in sections 4.4.1 and 4.4.2 respectively. This section explores the influence of the protocol on IFIS designs when subjected to fault conditions using an example architecture. This is done to discover if the expectations described in section 4.2.1 are fulfilled. Figure 4-7 shows the architecture of the example used to explore the behaviour of the existing protocols under fault conditions.

Figure 4-7 shows a shift register of the form described in Figure 4-6. The shift register has coded data input at **A** and coded data output at **Z**. Furthermore, the intermediate element outputs are observable and are labelled according to the parent element names.



*Figure 4-7 Architecture used to demonstrate fault conditions.*

In Figure 4-7 the dotted line shows the feedback from the element labelled **E1** as being separate from the element output. This is also the case for the remaining elements in the shift register and more typically represents practical design implementations. This can be attributed to the difference in output buffering required as a result of fanout in complex designs. Furthermore, the clock line has been omitted from Figure 4-7 for clarity. The fault site is marked by  $\alpha$  and is positioned such that the effects are not observable at **E1** but are observable by element **E2**. Such a fault may be caused by a defect in the physical interconnect such as a crack (intermittent behaviour) or may result from metal migration. The following discussions describe behaviour resulting from the injection of a  $1 \rightarrow 0$  fault injection and simulations of behaviour under such fault conditions are presented where necessary. The generic

behaviour of the design is unaffected by the coding scheme. The saturated coding scheme described in section 4.3 is employed for the purpose of example.

**Elastic protocol:** According to the expectations from IFIS described in section 4.2.1, self-transitions resulting from the processing of upstream elements signify an error. Furthermore, once an error has occurred the protocol must prevent future data propagation. The introduction of a temporary logic affecting fault has the effect of temporarily halting data flow within the shift register because the protocol condition is not fulfilled. However, once the fault is removed, it is possible for new data to be correctly represented at the original fault site. Following data loss processing resumes according to the behaviour described in Figure 4-4. The elastic protocol is therefore incapable of ensuring permanent prevention of data propagation following fault manifestation.

**Inelastic protocol:** Figure 4-8 shows a simulation of the example design under fault conditions.

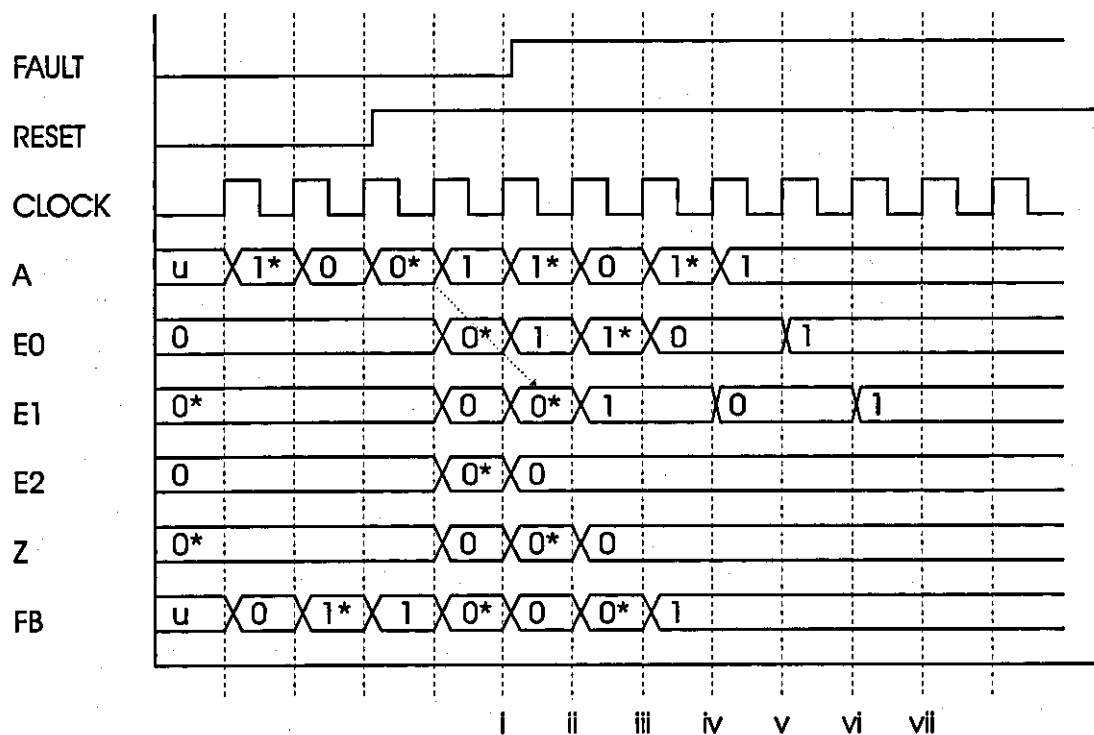


Figure 4-8 Effects of a specific fault on the behaviour based on the existing inelastic protocol.

The fault is activated between the rising clock edge labelled *i* and that labelled *ii* following normal design initialisation using the RESET signal. The fault becomes apparent at the output to element E2 at the edge *ii*. This is because the input to E2 is not E1 (0\*) due to the fault  $\alpha$ . Element E2 observes a 0 from E1 and a 0\* from element Z. This does not fulfil the existing inelastic protocol condition and so E2 does not process. The halting is shown to propagate to elements Z and E1 at edge *iii* and to element E0 at edge *iv*. At the edge *iv* element E1 processes. This is because the stable, halted, state of E2 belongs to the same state set as the most recent state held on the output of element E0. The resultant transition at output E1 (1 to 0) is illegal according to table 4-1. Hence, this protocol condition does not satisfy the requirements defined in section 4.2.1.

The reason that this behaviour occurs is that the difference between the initial halting of the predecessor element and the successor element (say, E0 and E2) is two clock cycles. Thus the outputs of these cells belong to the same data set and therefore fulfil the protocol condition for the current element.

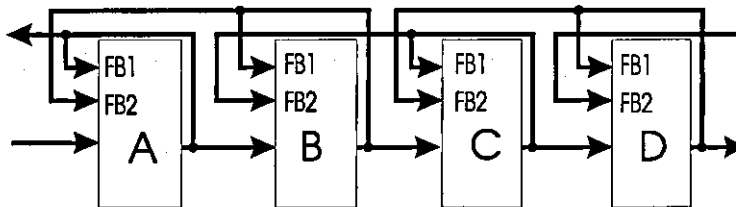
#### 4.5.2 Proposed Protocol Condition

In section 4.5.1 it was found that neither the elastic nor the inelastic protocol could be guaranteed to completely prevent error recovery. They both exhibit the potential to generate corrupt data. This is because recovery is inherent in the elastic protocol and because an insufficient number of node values were considered by the inelastic protocol. A modification to the inelastic protocol is presented and the behaviour is analysed.

Examination of Figure 4-8 shows that the initial halt of all elements results in all element outputs holding data from a single set (Set 0 or Set 1). This follows from halting being propagated at the rate of one element per clock cycle. Clearly, for fault-free operation, the outputs of neighbouring elements always belong to different code sets. To check this requires the monitoring of direct neighbours. The existing protocol conditions do not perform this check.

To address the problem highlighted in section 4.5.1 a new protocol condition is proposed. The behaviour of the current cell according to the proposed protocol is described in the following way.

Iff (  $(S(p) = \text{Set } x)$  and  $(S(c) = \text{Set } \bar{x})$  and  $(S(s) = \text{Set } x)$  )  
 then  
     next  $S(c) = f(S(p))$   
 else  
     next  $S(c) = S(c)$



*Figure 4-9 Interconnection of processing elements 'A', 'B', 'C', and 'D' according to the proposed protocol condition.*

Figure 4-9 shows the implication of the proposed protocol condition. The proposed protocol condition requires that processing elements monitor their own outputs thus increasing the number of inputs to each element. Figure 4-9 shows the additional self-feedback as **FB1** while the existing successor output supplies **FB2**.

#### 4.5.2.1 Fault-free conditions

The proposed protocol condition is now applied to the example introduced in section 4.5.1. For clarity, the signals labelled **FB1** in Figure 4-9 can be considered as internal to each processing element, thus the architecture shown in Figure 4-7 still applies.

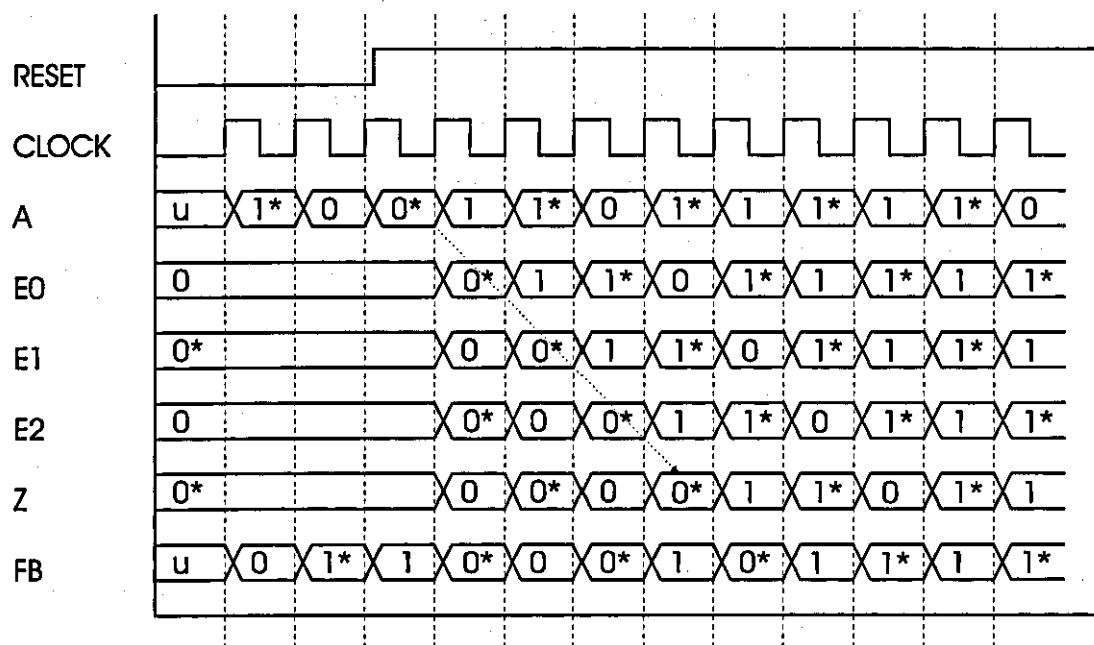


Figure 4-10 Behaviour of the proposed protocol under fault-free conditions.

Comparison of the inelastic protocol condition in section 4.5.1 and the proposed protocol condition in section 4.5.2 reveal that under fault-free conditions, both the behaviour and symbol rate are identical. Examination of Figure 4-10 shows this to be the case. Figure 4-10 shows that under fault-free conditions the data progresses through the pipeline at the rate of one element per clock cycle.

#### 4.5.2.2 Faulty conditions

Despite identical behaviour between the inelastic protocol condition in section 4.5.1 and the proposed protocol condition in section 4.5.2 under fault-free conditions, the behaviour of designs implemented according to the different protocol conditions can differ under fault conditions.

In Figure 4-11, the fault is enabled between the rising clock edges labelled *i* and *ii*. The cell with output E2 cannot process on the clock edge *ii* because the fault modifies E1 causing its parity to be incorrect. Furthermore, unlike the scenario shown in Figure 4-8, element E1 cannot process on the clock edge labelled *iv*. While the outputs from

element E0 and E2 conform to the protocol condition, the monitored output of E1 itself is not from the correct data set. The protocol condition is therefore not fulfilled.

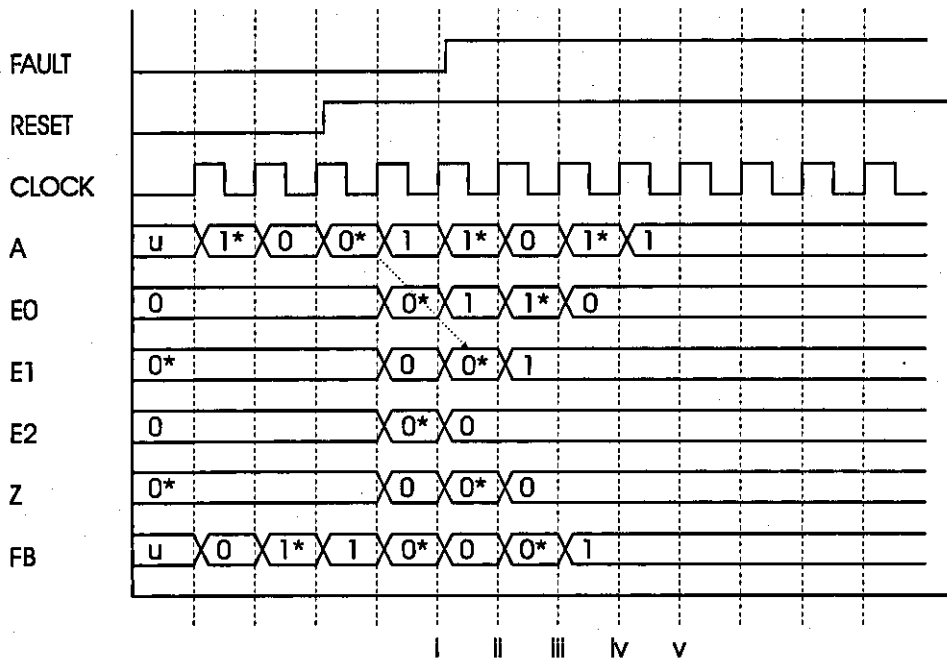


Figure 4-11 Behaviour of the proposed protocol under a single injected fault condition.

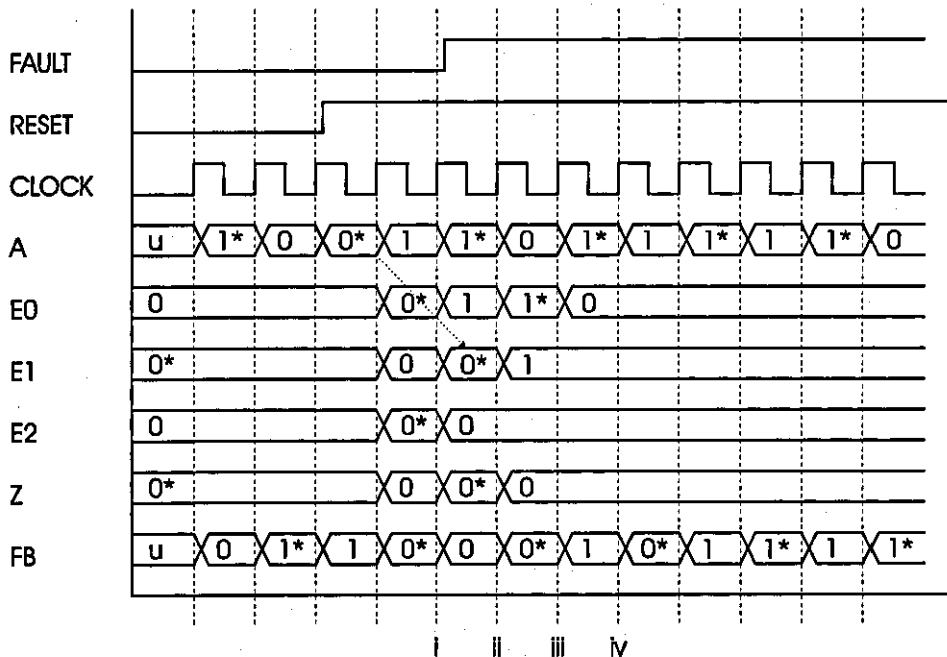


Figure 4-12 Behaviour of the proposed protocol under a single injected fault-condition and where the code translation blocks ignore the effects of error detection.

Figure 4-12 shows that the pipeline remains frozen even when data is continually applied to the upstream/downstream pipeline inputs. This situation can occur if the circuit external to the pipeline ignores the halting information passed to it by the pipeline itself. This may result from additional faults or from design techniques not conforming to the specification.

#### 4.5.3 Summary of Behaviour of Alternative Versions of IFIS

The behaviour of a potential on-line test system based on a commonly used asynchronous (elastic) protocol under faulty conditions was described in addition to the generic protocol under normal conditions. Furthermore, an inelastic protocol based on monitoring the same number of nodes in the same logical position was examined. Neither of these protocols satisfied the behavioural requirements specified in section 4.2.1. A modification to the inelastic protocol was proposed and the behavioural attributes examined under fault-free and fault conditions. While the example only highlighted a specific fault type, a more in depth examination of behaviour under different fault conditions applied to a design implemented using the proposed protocol condition is performed in chapter 7.

Table 4-3 summarises the behavioural properties associated with the different protocol/coding techniques.

Protocol/code combination	Maximum symbol rate	Maximum data rate	Halting type (worst case)
Elastic-RTZ	$\frac{1}{2}$ clocks	$\frac{1}{4}$ clocks	Temporary
Elastic-Saturated	$\frac{1}{2}$ clocks	$\frac{1}{2}$ clocks	Temporary
Inelastic-RTZ	1/clock	$\frac{1}{2}$ clocks	Temporary
Inelastic-Saturated	1/clock	1/clock	Temporary
Proposed-RTZ	1/clock	$\frac{1}{2}$ clocks	Permanent
Proposed-Saturated	1/clock	1/clock	Permanent

*Table 4-3 Behaviour of the protocol/coding combinations.*



Table 4-3 shows that when considering data rate and halting behaviour, the proposed inelastic protocol using the saturated coding exhibits the most attractive properties within the context of on-line testing.

#### 4.6 TEST COVERAGE ASSOCIATED WITH ALTERNATIVE VERSIONS OF IFIS

The maximum test coverage that can be achieved for a specific design is restricted by its structure and the test vectors which can be applied to it. By definition, a protocol restricts the vector set which can be processed by processing elements and therefore potentially restricts the maximum test coverage achievable for a specific design.

The proportion of codes which allow processing is dependent on the coding scheme used and the protocol in addition to the width of data to be carried by the code. Table 4-4 contains data relating to existing RTZ code/protocol combinations designed to transfer data of width one.

Data code	Feedback code	RTZ Legal State	RTZ Elastic Process	RTZ Inelastic Process
00	00	√		√
00	01	√	√	
00	10	√	√	
00	11			
01	00	√	√	
01	01	√		√
01	10	√		√
01	11			
10	00	√	√	
10	01	√		√
10	10	√		√
10	11			
11	00			
11	01			
11	10			
11	11			
<b>Total</b>		9	4	5

Table 4-4 Characteristics of existing protocol conditions on a single data bit RTZ code.

All possible binary combinations obtained by concatenating the data code and feedback code are represented in table 4-4. Those combined words that do not contain the illegal RTZ code described in section 4.3.1 are marked in the **RTZ legal state** column. The markers in the **RTZ Inelastic Process** and **RTZ Elastic Process** column correspond to combinations that fulfil the appropriate protocol conditions.

Protocol/code combination	Number of state codes	Number of possible binary codes	Number of codes which cause processing
Elastic-RTZ	$3^{(d+1)}$	$2^{2(d+1)}$	$2^d + 2$
Elastic-Saturated	$2^{2(d+1)}$	$2^{2(d+1)}$	$2^{(d+2)}$
Inelastic-RTZ	$3^{(d+1)}$	$2^{2(d+1)}$	$2^{(d+1)} + 1$
Inelastic-Saturated	$2^{2(d+1)}$	$2^{2(d+1)}$	$2^{(d+2)}$
Proposed-RTZ	$3^{(d+2)}$	$2^{2(d+2)}$	$2^{(d+1)} + 1$
Proposed-Saturated	$2^{2(d+2)}$	$2^{2(d+2)}$	$2^{(d+2)}$

Table 4-5 Characteristics of code/protocol combinations carrying  $d$  data bits.

Table 4-5 describes the characteristics of the code/protocol combinations presented in this chapter. In the table  $d$  represents the width of the data applied. Table 4-4 applies to RTZ codes where  $d$  is 1. The information contained in Table 4-5 is graphically represented in Figure 4-13 and Figure 4-14.

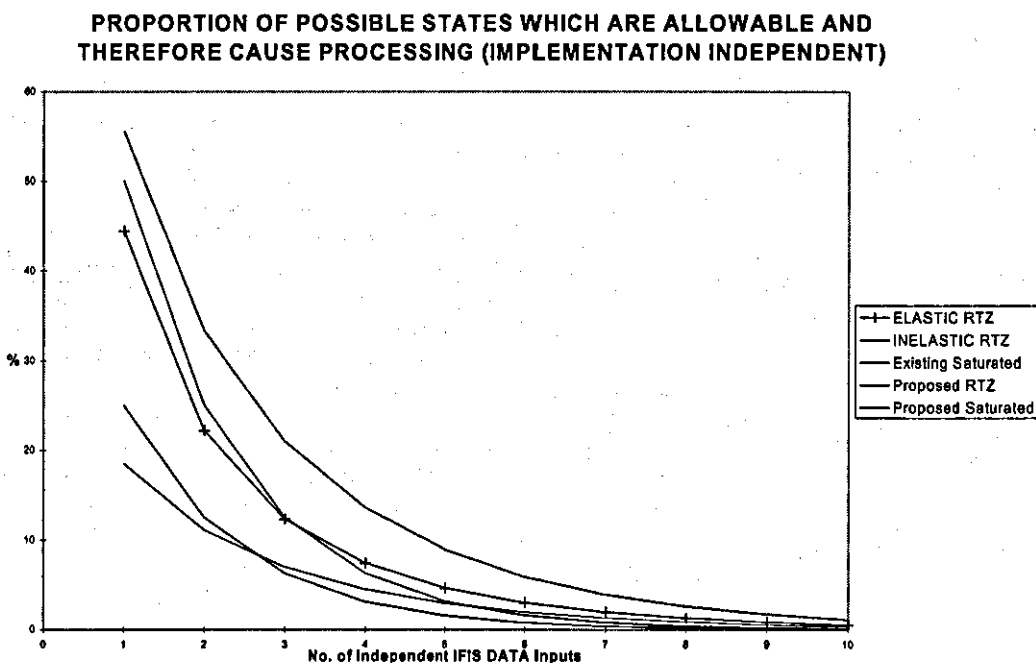
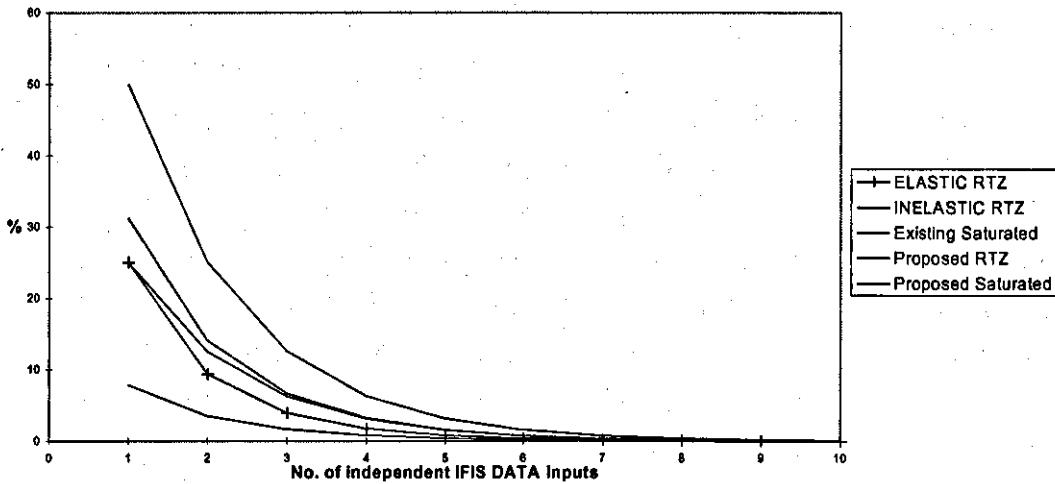


Figure 4-13 Proportion of states that allow processing in different code/protocol combinations and with different input data widths.

**PROPORTION OF POSSIBLE STATES WHICH ARE ALLOWABLE  
AND THEREFORE CAUSE CELL PROCESSING (CONVENTIONAL  
DIGITAL IMPLEMENTATION)**



*Figure 4-14 Proportion of binary state codes that allow processing in different code/protocol combinations and with different input data widths.*

Figure 4-13 shows the proportion of states for each coding/protocol combination that fulfil the appropriate protocol condition. Figure 4-13 shows that the inelastic protocol with RTZ coding appears to be the least restrictive of those examined. The information presented in Figure 4-13 is independent of implementation, and therefore ignores the illegal state that is inherent of providing three separate states using a dual-rail coding scheme.

Figure 4-14 shows the proportion of protocol condition fulfilling codes for each coding/protocol combination. The information contained in Figure 4-14 is specific to binary implementations and is therefore more relevant to the remainder of the work presented in this thesis. In Figure 4-14, the existing saturated codes are the least restrictive of the schemes examined. Furthermore, as the data width increases, the difference in potential restriction between all of the considered coding/protocol combinations becomes indistinguishable.

## 4.7 CONCLUSIONS

Evidence has been presented showing that data throughput; behaviour under fault conditions and the potential to restrict on-line test coverage can all be influenced by the choice of coding scheme, type of protocol and choice of monitored nodes.

Three sets of processing rules (protocols) were identified. Elastic and inelastic protocols that required the monitoring of predecessor and successor outputs were both found to be incapable of ensuring permanent halting after error detection. This is inherent for elastic protocols because the states of monitored nodes do not change concurrently, thus an unstable relationship between them exists. In the case of the inelastic protocol this behaviour could be attributed to insufficient monitoring. This was confirmed by comparison with the third protocol: an inelastic protocol that incorporates an additional condition (monitored node).

Data throughput is affected by the choice of protocol type (elastic/inelastic) and the coding scheme. In the case of inelastic protocols all symbols move in unison. Consequently, the symbol rate is the same as that for conventional binary systems. The elastic protocol only allows symbols to progress following acknowledgement of their previous use. This requires a minimum of two processing cycles to achieve and therefore elastic protocols are limited to a maximum of half the processing rate of that exhibited by conventional binary systems. Coding can limit the data throughput when the symbol rate does not represent the data rate. The RTZ coding scheme separates data with tokens and therefore falls into this category.

The potential to restrict on-line test coverage is caused by restricting the ratio of codewords to non-codewords, thus restricting the range of vectors that can be used for on-line fault stimulation. Evidence resulting from comparisons of the three protocols shows that the additional protocol condition further restricts the ability to process, thus restricting the ratio of codewords to non-codewords. Comparison of the coding schemes shows that the RTZ coding scheme has a greater potential to restrict on-line test coverage because of its lower ratio of codewords to non-codewords. Furthermore,

due to the imbalance in the RTZ coding scheme, the choice of protocol can further influence this ratio.

Where increased test coverage depends on the application of specific vectors to the processing element under scrutiny, the more restrictive coding/protocol combinations are least likely to permit the specific vector application. While the potential constraints of the protocol/coding combinations indicate that test vectors can only be chosen from a subset of all possible input vectors, this does not necessarily constrain the test coverage exhibited by the chosen vectors. The test coverage exhibited by applicable vectors depends on the suitability of the vectors to the circuit under test and the structure of the circuit.

Based on the previous discussion in the context of the expectations presented in section 4.2.1, the proposed protocol condition in conjunction with saturated dual-rail coding offer the most attractive properties. This combination is therefore selected as a basis for the design of IFIS systems.

# **CHAPTER FIVE**

## **IFIS CELL DESIGN**

### **5.1 OBJECTIVES OF CHAPTER**

The choice of IFIS protocol directly influences the interconnectivity of processing elements (cells) within IFIS systems. The protocol was selected in chapter four. This investigation is targeted at the internal circuit structure of the IFIS cells that support the chosen protocol and dual-rail code. The objectives of this investigation are to:

- Understand the influence of applying different design techniques to the generation of reference information and verification/error management circuitry within IFIS processing elements.
- Select that structure which provides the best features in the context of on-line testable systems. The selected structure will then be adopted as a template for future designs.

### **5.2 INTRODUCTION**

The structure of designs is known to affect the properties associated with them. Within the context of design-for-test, circuit structure is tailored to provide the most acceptable combination of test coverage and hardware penalty for the specific application. This investigation examines the influence of the circuit structure applied to IFIS processing elements on the following:

- Testability
- Data throughput
- Complexity

These attributes were chosen because they are those which are commonly used to distinguish between design for test techniques. Cell partitioning is considered and a number of alternative structures that were described in the review are applied to processing element sub-blocks. The options, together with alternative proposals are quantitatively compared and a generic IFIS processing element architecture is developed.

### 5.3 CELL PARTITIONING

Each IFIS processing element (cell) must be capable of providing reference data, performing validation and contributing to error management. Error management results from validation of incoming data to each IFIS cell. According to the protocol selected in chapter four, IFIS cell outputs are updated or they retain their previous value depending on the validity of incoming data. This output data is transmitted to other cells which in turn validate it. From the above discussion the processes of computation and transmission are separate tasks which can be performed in parallel. Figure 5-1 shows the generic cell structure implied by the above description.

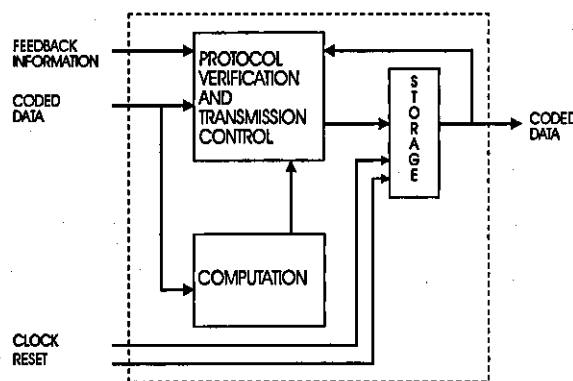


Figure 5-1 Anatomy of an IFIS cell.

In Figure 5-1 the cell boundary is marked by a dotted line. All signals, with the exception of the clock and reset signals are represented using dual-rail codes of the form chosen in chapter four. The clock and reset signal are carried by single wires in the conventional sense.

Figure 5-1 shows that the IFIS cell comprises three sub-blocks that perform the functions of protocol verification and transmission control, computation and storage. The functions of protocol verification and transmission control are performed within a single sub-block that controls which data codes are to be transmitted according to the validity of the codes appearing at the cell inputs. The computation sub-block operates on data codes appearing at the IFIS cell inputs and generates IFIS coded outputs. The computation block generates codes that represent a combinational function of the data content of the codes appearing at the cell inputs. The storage sub-block is updated on the rising edge of each clock cycle with data supplied at its inputs. The sub-blocks of interest within the scope of this investigation are the computation block and the block responsible for transmission control. The remainder of this investigation is structured to reflect this partitioning.

#### **5.4 COMPUTATION**

In Chapter two a number of techniques were described which permit reference data to be calculated for the purpose of validation. These techniques were classified as being information redundant, hardware redundant or time redundant. The techniques described were not specifically intended for use within a system that is controlled by a strict protocol like IFIS, and consequently modifications to the techniques need to be made to permit their use.

The generic modifications required to calculate IFIS codes are described and a representative design technique from each redundancy category is presented. The effects of design scaling in association with these techniques is assessed with respect to the attributes of testability, data throughput and complexity.

##### **5.4.1 Adapting Conventional Techniques to IFIS**

It is the task of the computation block to accept dual-rail codes and generate dual-rail codes. The dual-rail codes appearing at the computational block/cell inputs represent



binary data words where each component bit is represented as an individual dual-rail code. The dual-rail codes conform to the saturated four-phase coding scheme described in chapter four.

Examination of the saturated coding scheme (shown in table 4-1) reveals that the  $t$  bit from the dual-rail pair  $(t,f)$  always contains the binary representation of the coded data. This implies that the data content of the generated output codes can be calculated by examination of the incoming  $t$  bits alone.

The computation block output codes must belong to the same code set as the current inputs. This is because the outputs of the computation block become the outputs of the current cell on the next rising clock edge if the protocol is obeyed. If the symbol movement within IFIS designs is to conform to the selected protocol, codes appearing at the current cell outputs must belong to a different code set from the codes appearing at the outputs of the predecessor/successor cells.

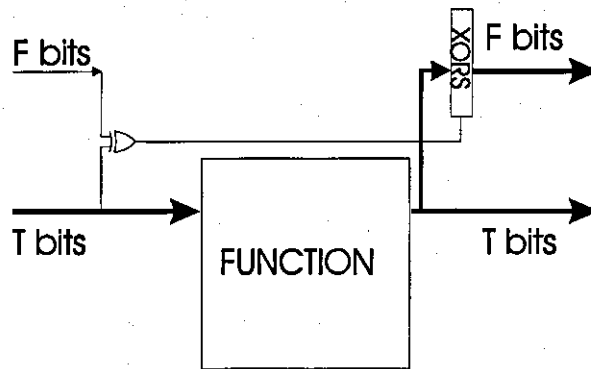
In chapter four the code sets were shown to be distinguishable by examination of the parity of member codes. This suggests that output codes can be generated by knowing which is the parent set of the input codes, and knowing the data content of the codes to be generated. While it is important that the output codes are valid according to the protocol, a number of techniques can be employed to generate them.

#### **5.4.2 IFIS Code Generation: The Need for Redundancy**

The discussion presented in section 5.4.1 suggests that the structure associated with accepting and generating  $t$  bits (henceforth known as the  $t$ -structure) is independent of the structure associated with accepting and generating  $f$  bits (henceforth known as the  $f$ -structure). Indeed, the  $t$ -structure is conventional in nature.

Figure 5-2 shows the behavioural architecture that generates correct output codes to satisfy the IFIS protocol under fault-free conditions but does not take fault effects into account. The  $f$ -structure monitors the parity of a selected dual-rail input data bit (all

should be from the same set) and ensures that the parity of all dual-rail outputs is the same.



*Figure 5-2 Code generation without computation protection.*

The architecture shown in Figure 5-2, while generating IFIS codes, does not generate the reference information independently from the data that it is supposed to protect. Indeed, the reference information is calculated from it thus ensuring the occurrence of fault masking under fault-conditions.

### 5.4.3 Relevant Design Properties

Unlike the context in which testability is normally applied, the on-line test environment provides error detection based on the comparison between data observed and reference data that was concurrently calculated. In the case of IFIS, the actual data and the reference data are combined to form IFIS codes at the output of the IFIS cell. The IFIS codes themselves are checked for correctness by other IFIS cells. The ability of codes to detect errors was defined in [Russell89] and is called the error detection ability (EDA). The EDA is applied to on-line error detecting techniques and as such assumes independence between generation of reference information and the data that it corroborates.

It is well known that for dual rail codes all single bit errors can be detected because the Hamming distance between codewords is 2. Therefore any error that is confined either to data or is confined to reference generation is detectable.

A scalable complexity approximation can be obtained for the architecture shown in Figure 5-2 in terms of **nand gate equivalents**. To enable comparison with other techniques while also providing an insight into design scaling, a function with known scalability characteristics was selected.

The function is an  $n \times n$  cellular array multiplier. The particular architecture is described in [Almaini89] and comprises  $n^2$  partial product **and** gates and  $n^2 - n$  full adders. For such a multiplier with  $2n$  outputs,  $2n+1$  exclusive-or gates are required to realise the structure shown in Figure 5-2. As a basis for future comparison the complexity characteristics associated with this design are presented in Table 5-1.

Structure	Nand Gate Equivalent
And Gate	1
Xor Gate	3
Full Adder	8
Multiplier	$(8+1)n^2 - 8n$
Top	$9n^2 - 2n + 3$

*Table 5-1 Complexity of an unprotected architecture for generating IFIS codes.*

Table 5-1 shows the nand gate equivalent of each component where  $n$  represents the number of bits in the multiplicand (multiplier). The nand gate equivalents shown are approximations based on the number of MOS transistors in the CMOS implementations of standard cells, where a nand gate comprises four transistors. The entry labelled **Top** describes the number of nand gate equivalents in the complete structure shown in Figure 5-2.

#### 5.4.4 Information Redundancy

The architecture shown in Figure 5-3 constructs valid IFIS codes if the coded function generates valid residue codes. If this is not the case, the exclusive-or gate supplied by the comparator attempts to force the parity of all IFIS cell dual-rail outputs to be incorrect. This implementation contains only combinational circuitry. The speed of the components need therefore only be sufficient to ensure that function outputs are stable within the duration of one system clock cycle.

The IFIS protocol can detect any logical fault that affects only one rail of each dual-rail output pair. This property is characteristic of dual-rail codes. Assuming a single fault multiplicity, the fault can either occur within the f-structure or within the t-structure.

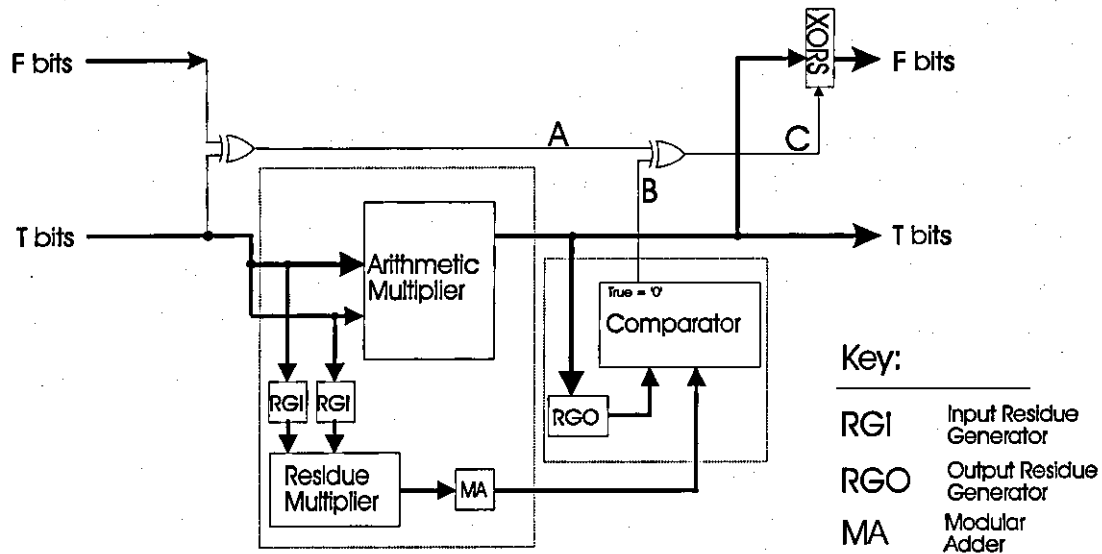


Figure 5-3 Code generation with computation protection provided by information redundancy

Under fault-free conditions the signal line 'A' should alternate on successive clock cycles because it carries the parity of dual-rail codes on the cell inputs. Any fault that affects 'A' either makes it assume an incorrect value or is insignificant. If 'A' is incorrect the signal 'C' becomes incorrect and therefore the cell outputs belong to the incorrect code set, as defined in Table 4-1. Any fault that is detectable by the residue coding scheme causes the signal line 'B' to rise. This causes 'C' to be inverted causing the cell outputs to be incorrect. If a fault occurs on 'B' such that it remains at zero, this is not detectable. For this reason, the ability of the function block to generate codes that accurately represent its internal integrity is less than the ability of the residue code to detect errors within the t-structure. The architecture exhibits the following additional limitations:

- Increasing the number of bits in the residue (to increase probability of error detection) decreases the suitability of the architecture. This is because the number of bits in the IFIS data inputs must be exactly

divisible by the number of bits in the associated residue prior to multiplication if a low cost residue code is to be used.

- Although it is possible to generate residue codes for Boolean algebra functions, the area overhead can be significant in comparison to the area overhead incurred for arithmetic functions [Russell89].

Table 5-2 shows the scalable nand gate equivalent of each component in the architecture with the exception of those defined in Table 5-1.

Structure	Nand Gate Equivalent
Arithmetic Multiplier	$9n^2 - 8n$
Residue Multiplier	20
Modular Adder (MA)	$3X_{\text{or}} + 8$
Residue Generator (RGi)	$((n-1)/2)MA$
Residue Generator (RGo)	$(n-1)MA$
Comparator	$4X_{\text{or}} + 2$
Top	$9n^2 + 38n$

*Table 5-2 Complexity of an architecture protected by information redundancy.*

The Table applies to a two check-bit residue code (base  $2^2-1=3$ ). The modulo adders, residue generators and comparator are therefore identical to those described in [Russell89].

#### 5.4.5 Hardware Redundancy

The architecture shown in Figure 5-4 uses hardware redundancy to generate reference information and data independently. As was the case in section 5.4.4, this structure is purely combinational and therefore the speed of the components need only be sufficient to ensure that function outputs are stable within the duration of one system clock cycle. The f-structure in Figure 5-4 is constructed such that it forms a self-dual function as described in [Shedletsky78]. The employed exclusive-or structure ensures that the values held at 'A' and 'B' are always identical under fault-free conditions. The same properties apply to the values at 'C' and 'D'.

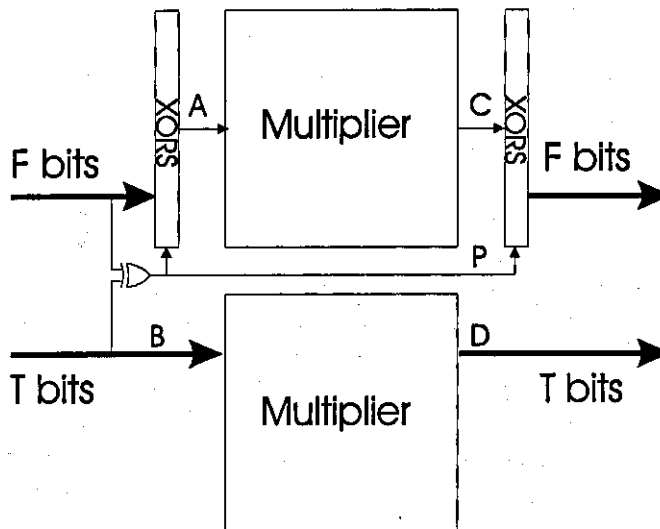


Figure 5-4 Code generation with computation protection provided by hardware redundancy.

Under fault-free conditions, the signal labelled 'P' alternates with each successive clock cycle. Any fault that causes 'P' not to be correct ensures a code from the incorrect code set to appear at the function outputs. The independence of the f-structure and the t-structure ensure that incorrect codes are formed at the function output under all excited fault conditions that affect only the outputs of the t-structure or the outputs of the f-structure. This architecture has the following advantages:

- The architecture is generic and can be directly applied to any Boolean function. Furthermore, being generic facilitates the design process.
- Test patterns generated for testing one internal function can be used to test the other identical function in parallel during production test. This minimises pattern generation time and minimises production test time.
- All checking is performed by the IFIS code checker, thus maintaining the conceptual design partitioning.

Structure	Nand Gate Equivalent
Multiplier	$9n^2 - 8n$
Xor	3
Top	$18n^2 - 4n + 3$

Table 5-3 Complexity of an architecture protected by hardware redundancy.

Table 5-3 shows the scalable nand gate equivalent of each component in the architecture shown in Figure 5-4.

### 5.4.6 Time Redundancy

The architecture shown in Figure 5-5 uses time redundancy to generate reference information and data independently.

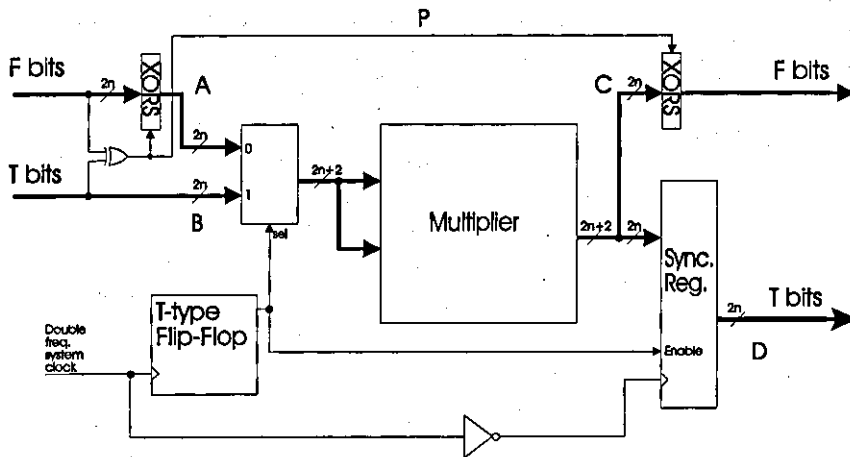


Figure 5-5 Code generation with computation protection provided by time redundancy

Unlike the architectures based on information redundancy and hardware redundancy, the components must be fast enough to perform calculations twice within the duration of a normal system clock cycle.

A similar exclusive-or structure is incorporated in the f-structure to that in Figure 5-4 such that under fault-free conditions 'A' and 'B' carry the same values. A similar observation holds for 'C' and 'D'.

The operation of the architecture shown in Figure 5-5 is based loosely on that described in [Patel83]. During the first calculation step the value held at 'B' is operated upon and the answer is stored in the synchronous register. Then the output of the synchronous t-type flip-flop toggles thus disabling the register and selecting the value held at 'A'. 'A' is processed through the multiplier using a different path to that used for processing 'B'. The shifted output of the multiplier appears at 'C' in time for the function outputs to be stable by the next rising system clock edge.

A description of fault-conditions affecting 'P' was described in section 5.4.5. In [Patel83] it was shown that for multiplication, a single bit shift in both the multiplicand and the multiplier is sufficient to ensure that fault masking does not occur. Consequently single faults are always detectable for the described multiplier. The following disadvantages are associated with this architecture:

- A local clock generator is required to produce a local clock for the function. This can cause testability problems by increasing the number of sequential elements, thus adding nodes with low levels of controllability and observability.
- Due to the requirement for storage, test pattern generation for such a block requires a sequential test pattern generator unless the synchronous register itself provides additional test access for use during production testing.
- The scaling of the arithmetic function outputs is dependent on the function. If the function is an adder, then only one dividing shift is required ( $11+11 = ((110 + 110) \text{ shifted once})$ ). However, 11 multiplied by 11 =  $((110 \times 110) \text{ shifted twice})$ .
- The same principle of operation cannot be applied to Boolean functions.

Structure	Nand Gate Equivalent
Multiplier $(n+1) \times (n+1)$	$9n^2 + 10n + 1$
Multiplexer	3
T-type Flip-flop	7
Register (width $2n$ )	$14n$
Top	$9n^2 + 42n + 18$

*Table 5-4 Complexity of an architecture protected by time redundancy.*

Table 5-4 shows the scalable nand gate equivalent of each component in the architecture shown in Figure 5-4.



### 5.4.7 Summary of Computation Techniques

A number of alternative computation block architectures have been presented and their characteristics have been discussed. Table 5-5 summarizes the most important characteristics of the alternative techniques in the context of IFIS computation block design.

Protection Method	Suitability to Arithmetic	Suitability to Non-arithmetic	Protocol Based SSA Coverage	Effect on Circuit Speed
None (behavioural)	√	√	Very low	~equal to dual
Residue coding	√	√*	Statistically high	< dual
Duality	√	√	100%	~1.0
Pipeline	√	X	100%	max. = 0.5

Table 5-5 Suitability of different architectural design approaches to the IFIS function.

All of the described techniques can be used for designing arithmetic computation blocks. However, residue coding is known to be inefficient for protecting logical (non-arithmetic) functions [Russell89]. This is characteristic of arithmetic codes.

Complexity comparisons for differing architectural implementations of an IFIS function based on cellular multiplication

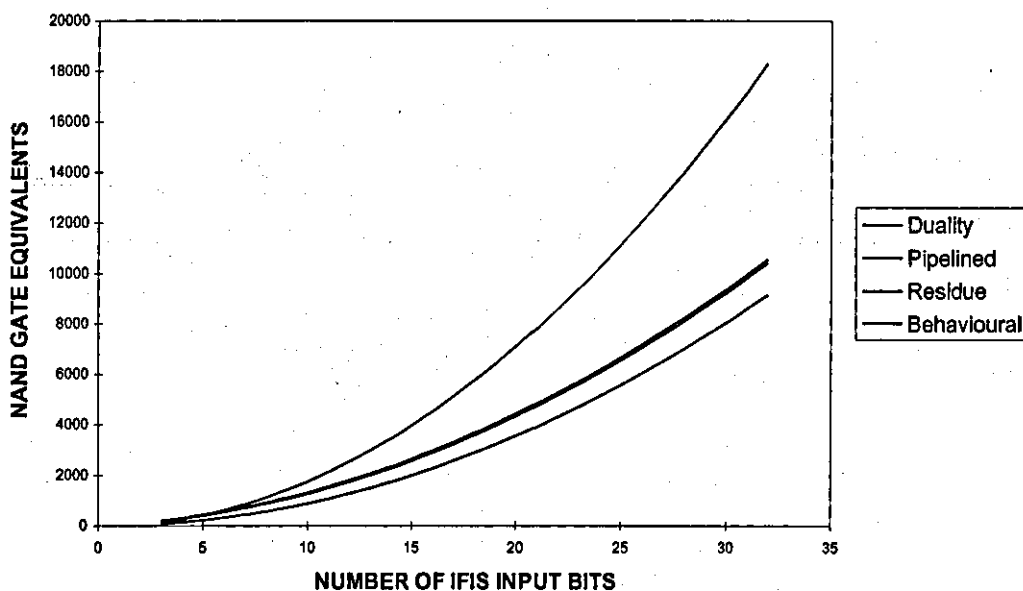


Figure 5-6 Complexity comparison of the techniques applied to computation block design.

While it was shown that non-arithmetic operations could be protected by the pipeline architecture used in [Patel83], the architectures were all regular in nature. This technique is not suitable for non-datapath Boolean functions.

The protocol based single stuck-at (SSA) fault coverage refers to the proportion of faults that cause protocol disruption. A qualitative approach has been applied because the values depend on the ratio of collapsed faults within a function block to the complete number of collapsed faults existing within the computation block (IFIS cell). The effect of different techniques on circuit speed is described with respect to a conventional equivalent implementation.

Figure 5-6 provides a graphical representation of the complexity data accompanying the descriptions of each design alternative. Clearly, the architecture based on hardware redundancy results in the most complex designs.

## 5.5 TRANSMISSION

The selection of codes for transmission to other IFIS cells is based on the adherence of cell inputs to the IFIS protocol. As such, the task of transmission control can be partitioned into the sub-tasks of data validation and data selection. This partitioning leads to independent circuits that perform the separate tasks. The partitioned design option implies a clear boundary between the sub-functions thus allowing the application of a number of standard design approaches to the task of data validation.

An alternative approach is to design a single unit that performs both tasks, without constraining the structure to contain a logical interface between them. This approach may lead to a more efficient design.

A number of alternative techniques for realising the different partitioning options are presented. The effects of design are assessed with respect to the attributes of testability, and complexity.

5.5.1 Partitioned Design

A number of design techniques are presented for the separate tasks of code validation and code selection. The techniques are compared with respect to complexity and single stuck-at fault coverage.

5.5.1.1 Detection alternatives

The task of the presented architectures is to check the validity of the IFIS codes appearing at the IFIS cell inputs and generate an output accordingly. Figure 5-7 shows the architectures that were chosen for comparison.

The decoder architecture shown in Figure 5-7(A) is that which is directly derived from the behavioural description of the validation circuitry and synthesised using the Intergraph tools. When the parity of the codes placed on the dual-rail data inputs and successor cell feedback are all *even* and the parity of the current cell output is *odd* or visa-versa, then the design output assumes a logic '1'.

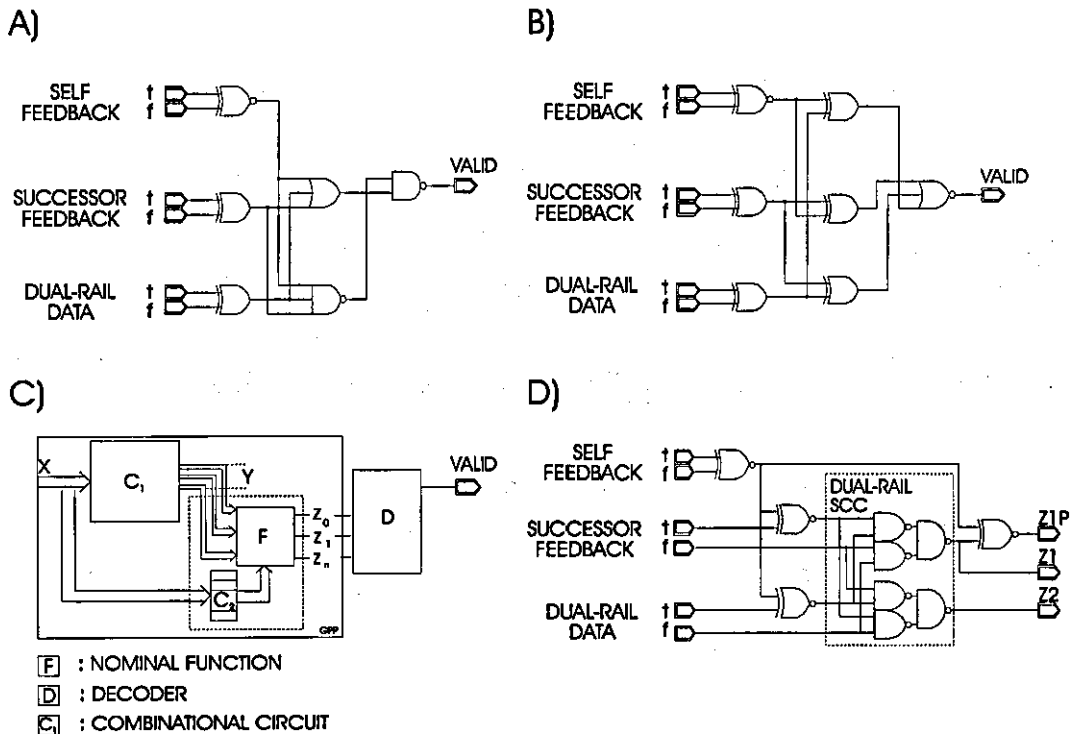


Figure 5-7 Protocol validation architectures. A) Decoder B) Exclusive-or tree C) Group Parity Prediction based protection D) Self-checking checker hybrid.

The exclusive-or-tree architecture shown in Figure 5-7(B) forms a comparator. Exclusive-or trees are known to be difficult to test under production test conditions because of the inability to control internal circuit nodes independently. However, the on-line test coverage afforded by IFIS codes applied to these structures is unknown. When the parity of the codes placed on the dual-rail data inputs and successor cell feedback are all the *same* and the parity of the current cell output is *different* then the design output assumes a logic '1'.

The ability of an on-line test system to test the checking circuitry in addition to the circuitry that it protects is evidently attractive. The IFIS validation circuitry is no exception. The architectures presented in Figure 5-7(C) and Figure 5-7(D) are targeted at fulfilling this goal.

The group parity prediction architecture shown in Figure 5-7(C) was presented in [Fujiwara84] and can be applied to any combinational design. It is not specifically targeted at IFIS and consequently is not shown with IFIS specific modifications. This technique checks the outputs of the combinational circuit ' $C_1$ ' against reference data provided by ' $C_2$ '. The relationship between  $C_1$  and  $C_2$  is such that when combined by the function 'F', members of a specific code set occur at 'Z', where  $Z = \{Z_0, Z_1, \dots, Z_n\}$ . This is definable at the time of design and determines the valid codeword inputs to the output decoder shown in Figure 5-7C.  $Z_0$ ,  $Z_1$  and  $Z_n$  are shown in Figure 5-7C. To access parity information within an IFIS code checker, the checker inputs can be checked directly. The only circuitry thus checked becomes an array of exclusive-or gates. To check this array ( $C_1$ ) requires that  $C_2$  is an array of exclusive-(n)or gates, which increases the circuit complexity by ~100%. However, the nature of IFIS code sequences ensures that if computation blocks are fully testable, then the parity checking gates within any IFIS checker can be fully exercised. If an incorrect output is generated by the IFIS checking circuitry then the characteristic halting takes over. The above discussion precludes further examination of this architecture.

The self-checking-checker hybrid shown in Figure 5-7(D) is based on a self-checking dual-rail checker described in [Siewiorek92]. The SCC expects and only generates

odd parity codes under fault-free conditions. The outputs Z1P and Z2 are additional IFIS cell outputs and thus must form correct IFIS codes under fault-free conditions. This increases the observability of the output of validation checking circuitry. The Z1 and Z2 outputs are used to control code selection. When the parity of the self-feedback input is even (odd), the parity of the remaining dual-rail inputs should be odd (even) and the Z1P and Z2 outputs should form an odd (even) parity code.

### a). Complexity

Once a specific checker architecture has been chosen, it must be possible to tailor it to the specific application.

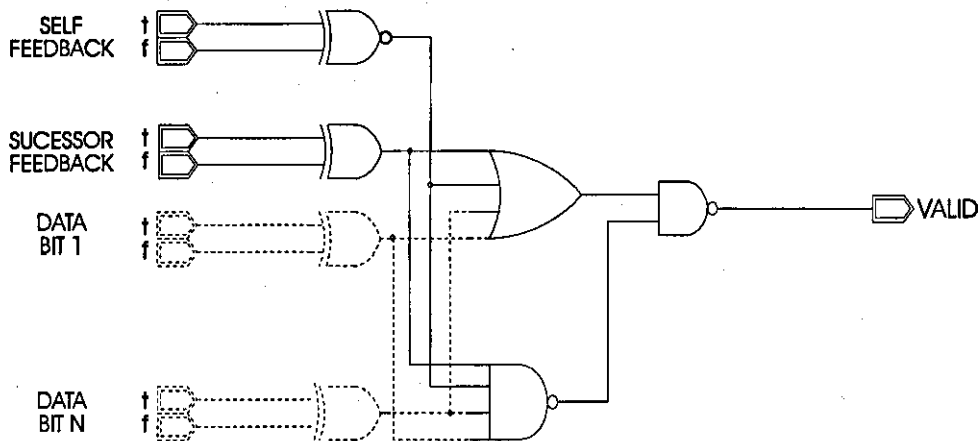


Figure 5-8 Generic decoder architecture.

In the case of the code checking circuitry for IFIS cells the number of coded data inputs to the IFIS cell dictates the complexity of the checker used to validate them. Figure 5-8 shows a generic implementation of the decoder that was introduced in Figure 5-7. This circuit is used as an example to describe how input scaling effects the complexity of the various architectural alternatives.

Figure 5-8 shows that irrespective of the number of dual-rail data inputs, the number of feedback inputs remains constant at 2. The generic architecture thus contains 'n+2' exclusive-or/exclusive-nor gates and one two-input nand gate. The complexity of the remaining two gates varies according to 'n' such that together they contain  $4(n+2) + 2$

transistors. This is equivalent to 'n+2' two-input nand gates and an inverter. Based on the nand gate equivalent for an exclusive-or/exclusive-nor gate used in Table 5-1, the total generic complexity associated with this architecture is  $(3+1)(n+2) + 1.5$  nand gate equivalents. Table 5-6 describes the effects of scaling 'n', the number of dual-rail data inputs, on the complexity of the examined architectures.

Architecture	Nand Gate Equivalent
Decoder	$4n + 9.5$
Exclusive-or tree	$(13n + 26) / 2$
SCC hybrid	$9(n+1)$

*Table 5-6 Complexity of alternative checkers.*

#### b). Test coverage

The maximum test coverage that can be exhibited by a design depends on the design structure and the range of vectors to which it can be subjected. Conventional synchronous designs are frequently subjected to structural tests during production testing. To simplify test application and response retrieval test structures are introduced that allow access to combinational circuit inputs and outputs. This is the scenario assumed when referring to production testing in this thesis.

In Chapter four, a potential restriction in the range of values appearing at the inputs of IFIS cells was identified. This restriction was due to the IFIS protocol. As such, the effects of this restriction can be quantified in terms of the maximum test coverage affordable by a set of protocol obeying (those occurring under fault-free conditions) vectors when applied to specific design structures. Furthermore, advantage can be taken of the halting properties of the IFIS methodology. By introducing an incorrect IFIS code, the halting mechanism can be exercised. This results in a cell freeze under true fault-free conditions; the only recovery being a system reset. Figure 5-9 shows a specific implementation of the decoder architecture which is used as an example to describe the effects of changing 'n' on the maximum single-stuck-at fault coverage achievable under the conditions described above. The inputs SFB\_T, SFB\_F, FB\_T, FB\_F, A\_T and A\_F combine to form the dual-rail self-feedback, successor feedback

and data inputs respectively.

In Figure 5-9 the positions and multiplicity of certain stuck-at faults have been labelled. For example, this design possesses six input lines with twelve possible single stuck-at faults associated with them. Generally, this equates to  $4(n+2)$  SSA faults. To represent the physical implementation of a design more than two SSA faults can be associated with a single logical node. This is the case for the outputs of gates 'U', 'V' and 'W'. The outputs of these gates can be directly associated with  $2(n+2)$  possible SSA faults. Due to the fanout of these nodes, additional SSA faults are associated with the inputs of gates 'X' and 'Y'. Fault collapsing results in the total number of faults described in Figure 5-9.

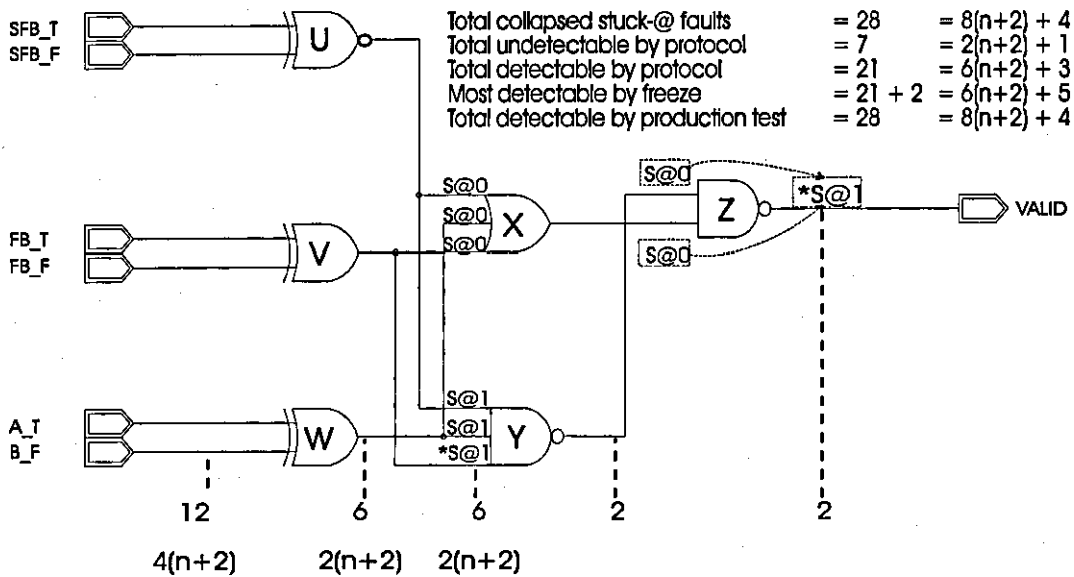


Figure 5-9 Test properties of the decoder architecture where  $n$  is equal to 1.

Protocol obeying vectors cannot simultaneously sensitise the paths from the output of cell 'X' to cell 'Z' and from cell 'Y' to cell 'Z'. Consequently the faults labelled at the inputs to cell 'X' and cell 'Y' are not testable by the protocol. Similarly, under fault-free conditions, the output of cell 'Z' never assumes a logic '0' and the stuck-at-1 fault associated with it is never tested. The total number of SSA faults not testable by protocol obeying vectors is thus  $2(n+2) + 1$ .

An incorrect code combination applied to the decoder inputs causes the outputs from

cell 'X' and cell 'Y' to be sensitised and an additional SSA fault on one of their respective inputs is tested. In this case a freeze was suggested as resulting in the vector (1,1,0) occurring on the cell outputs (U,V,W). The SSA fault tested by this specific occurrence is marked with an asterisk in the figure. When no fault exists within the decoder, the inputs to cell 'Z' assume a logic '1' and the decoder output assumes a logic '0', thus testing it for a stuck-at-1 condition. An induced freeze thus tests for an additional 2 faults.

Further examination of Figure 5-9 shows that the faults undiscovered by the protocol followed by a freeze are located at the inputs to two gates namely, gate 'X' and gate 'Y'. Only one of these faults is tested by the freeze condition. An approach that could reduce the need to check these faults could be to implement the complete checking structure as a standard cell and design the connections to these nodes defensively. While still being testable during production testing, the requirement to continually check them in the on-line test environment would be reduced. This technique would not be suitable for application to all nodes because it invariably increases the physical area of the standard cell and could become prohibitive. The effects of applying this method are described within Table 5-7 in the row labelled **\*Decoder**.

Architecture	Collapsed SSA Faults	Production Test	Protocol	Protocol + Freeze
Decoder	$8(n+2)+4$	$8(n+2)+4$	$6(n+2)+3$	$6(n+2)+5$
* Decoder (std_cell)	$8(n+2)+4$ $[6(n+2)+4]$	$8(n+2)+4$	$6(n+2)+3$	$6(n+2)+4$
Exclusive-or tree	$11(n+2)+2$	$10(n+2)+2$	$10(n+2)+1$	$10(n+2)+2$
SCC hybrid	$30n+22$	$30n+22$	$30n+22$	$30n+22$

*Table 5-7 Maximum SSA test coverage for alternative checkers.*

Table 5-7 describes the total number of collapsed stuck-at faults and the maximum coverage under different test conditions in terms of n: the number of dual-rail data inputs to the IFIS cell. These descriptions were obtained by applying the techniques described in the above example to the examined architectures.



### 5.5.1.2 Data-flow Control

The task of the presented architectures is to supply the storage block within the IFIS cell with its own, current output, or with the output of the computation block. Figure 5-10 shows the considered architectures.

The simple multiplexer array, shown in Figure 5-10(A), accepts a single control signal (valid signal) which when high causes the selector to route the output of the computation block to the IFIS cell output. The valid signal is that which is generated by selected architectures described in section 5.5.1.1.

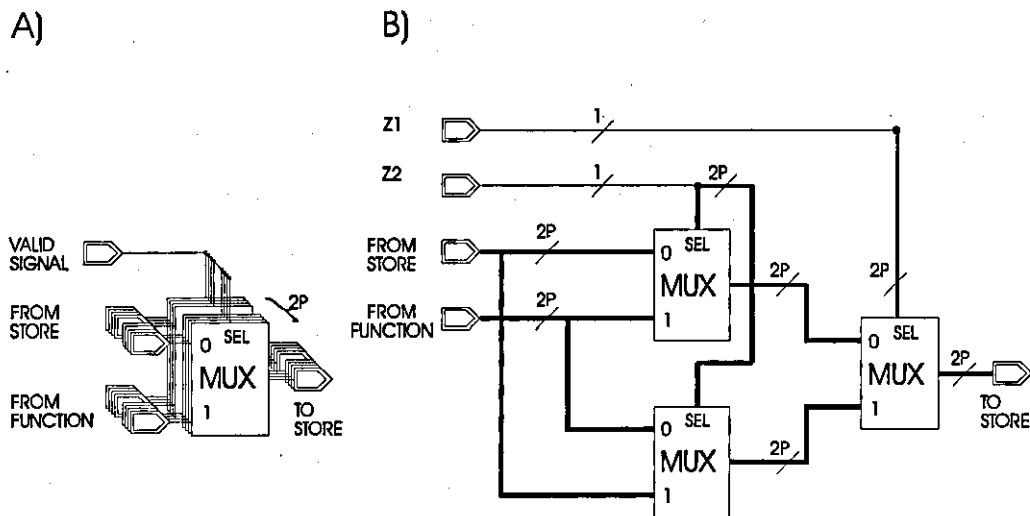


Figure 5-10 Selector architectures for selecting one of two busses, each containing  $p$  dual-rail codes. A) Simple multiplexer array. B) Selector controlled by a dual-rail code.

The selector controlled by a dual-rail code, shown in Figure 5-10(B), causes data to be routed from the computation block output to the IFIS cell output when a dual-rail code of odd parity appears on the control lines 'Z1' and 'Z2'. This architecture is suitable for use with the SCC hybrid shown in Figure 5-7. The Z1 and Z2 outputs from the SCC hybrid are connected directly to the Z1 and Z2 inputs of the selector respectively.

The data contained in Table 5-8 summarises the effects upon complexity, total number of collapsed stuck-at faults and maximum number of detectable faults for each of the examined selector architectures. This information was obtained in the same way as previously described.

Architecture	Complexity	Collapsed SSA Faults	Production Test	Protocol	Protocol + Freeze
Conventional multiplexer	6p	16p [+2]	16p [+2]	10p [+1]	14p [+2]
Multiplexer for SCC guard	18p	48p [+4]	48p [+4]	36p [+4]	42p [+4]

*Table 5-8 Characteristics of the selector units.*

In Table 5-8 'p' represents the dual-rail data width of the computation block output. The selector architectures connect to the outputs of validation circuitry. As such, the number of combined collapsed stuck-at faults is not accurately represented by summing those in each circuit because the connected nodes would be duplicated. The numbers within square brackets refer to the additional faults that must be considered if each architecture is to be considered as stand-alone.

### 5.5.2 Non-partitioned Design

These design alternatives combine the functionality of code validation and selection into a single design block. This design approach does not impose a conceptual interface between the functions and therefore does not inflict artificial constraints on the circuit. The need to impose a freeze on the IFIS system to test the single communication path between the otherwise separate functions may therefore be eliminated.

Figure 5-11 shows a single architecture that exhibits the functionality described. In Figure 5-11A the top level architecture is shown, while Figure 5-11B shows the selector block. The self-feedback and successor feedback are shown as 'SFB' and 'FB' respectively. The data inputs, labelled 'DN' in Figure 5-11A, also connect to the control input labelled 'CNTRL(N+2)' in Figure 5-11B.

The architecture selects data codes from the computation block or the storage output depending on the codes that it receives. The computation block outputs are selected if the parity of the IFIS cell data codes and the parity of the successor cell feedback are all even while the parity of the current cell output is odd, or visa-versa. Otherwise, the

current cell output is selected.

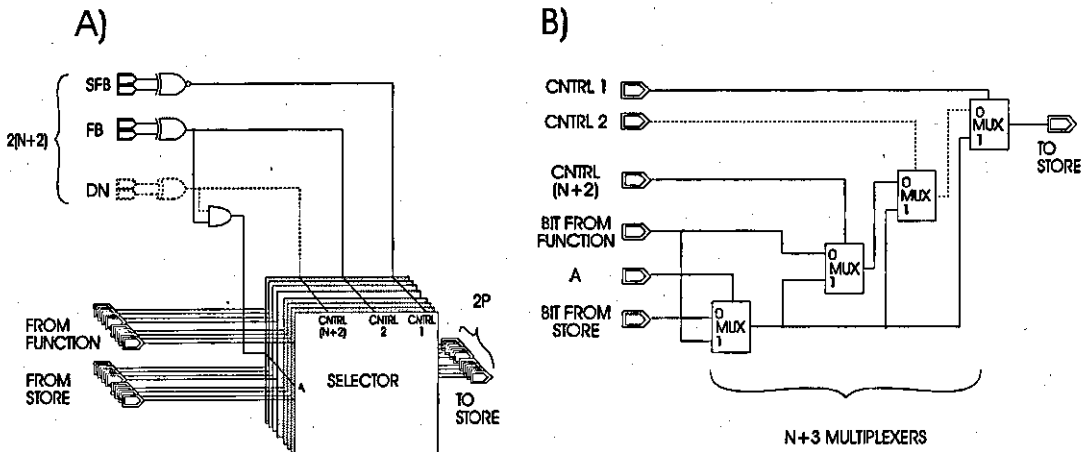


Figure 5-11 Combined decoder/selector architecture.

Examination of Figure 5-11B shows that under fault-free conditions, when the parity of 'DN' is even, the path from the function to the selector output via the '0' multiplexer inputs is taken. Conversely, when the parity of 'DN' is odd then 'A' is logic level '1' and the route via the '1' input of the output multiplexer is taken. Under all fault conditions the storage output is selected.

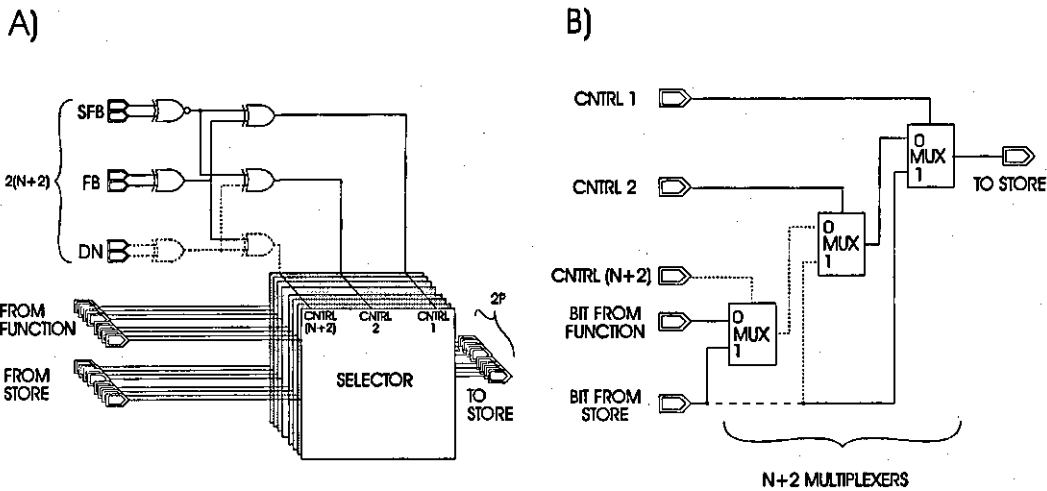


Figure 5-12 Combined comparator/selector architecture.

Figure 5-12 shows an alternative architecture based on comparison rather than decoding. The labelling convention adopted in Figure 5-11 is also used in Figure 5-12. Figure 5-12A shows the top level architecture and Figure 5-12B shows the structure of the selector within Figure 5-12A. Under fault-free conditions, the parity of 'DN' is the

same as the parity of 'FB' and is different from that of 'SFB'. This results in all signals with the prefix 'CNTRL' being set to zero, thus routing the computation block outputs to the store.

Architecture	Complexity	Collapsed SSA Faults	Production Test	Protocol	Protocol + Freeze
Decoder / selector	$3(n+2) + 6p(n+3) + ((n+1)/2)$	$12pn + 9n + 48p + 17$	$12pn + 9n + 48p + 17$	$6pn + 7n + 36p + 14$	$6pn + 17n + 38p + 14$
Comparator / selector	$6(n+2) + 6p(n+2)$	$12pn + 12n + 24p + 24$	$12pn + 12n + 24p + 24$	$6pn + 8p + 11n + 22$	$8pn + 12p + 11n + 23$

*Table 5-9 Characteristics of the combined checker/selector architectures.*

The characteristics of the combined architectures were obtained using the same techniques as those previously described and are contained in Table 5-9.

### 5.5.3 Summary of Transmission Control Techniques

A number of different architectures have been presented and their characteristics described. To compare the design techniques in a quantitative way requires that a relationship be established between the number of dual-rail data inputs ( $n$ ) and the number of dual-rail outputs from the computation block ( $p$ ). Clearly not all Boolean functions can be scaled. However, the multiplier example used in section 5.4 has  $2n$  inputs and  $2n$  outputs. For the purposes of comparison, it is therefore assumed that the data transmission controllers are part of an IFIS cell containing such a multiplier structure.

Figure 5-13 combines data from Table 5-6 to Table 5-9 inclusive and shows a comparison of the effects of scaling  $n$  on design complexity. In Figure 5-13 the complexities of the combined architectures, namely 'DEC\_MUX' and 'CMP\_MUX', are shown not to scale favourably. Conversely, the partitioned architectures scale in a similar way to one another in a linear fashion.

### Complexity Comparisons for Data Transmission Controllers

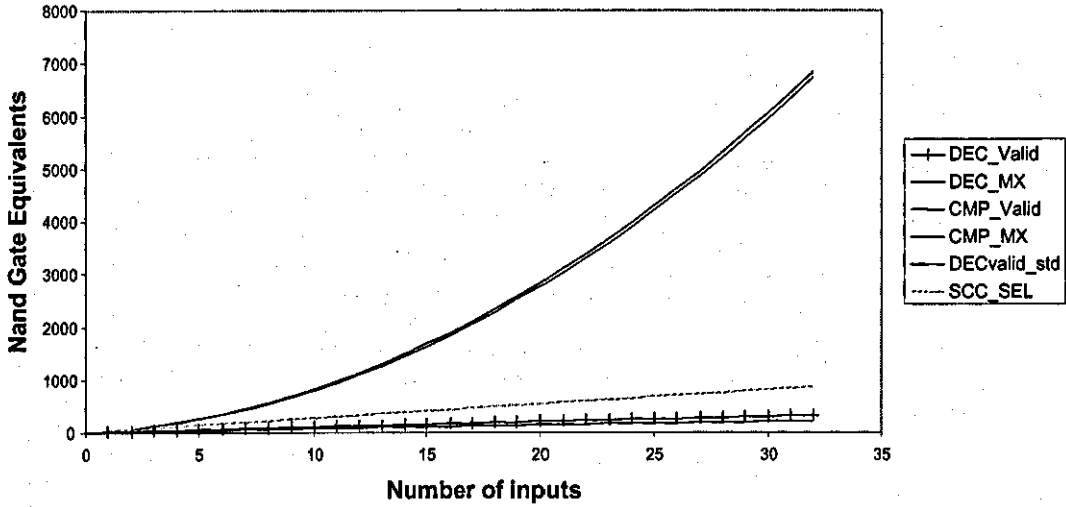


Figure 5-13 Scaling of design complexity for the alternative architectures.

Figure 5-14 follows the same labelling convention as that adopted in Figure 5-13. The maximum single stuck-at fault coverage for architectures based on comparison is shown to be limited to values of less than 100%. This is because of the inability to control internal circuit nodes independently even when access to design inputs is not restricted.

### Maximum Production Test SSA coverage

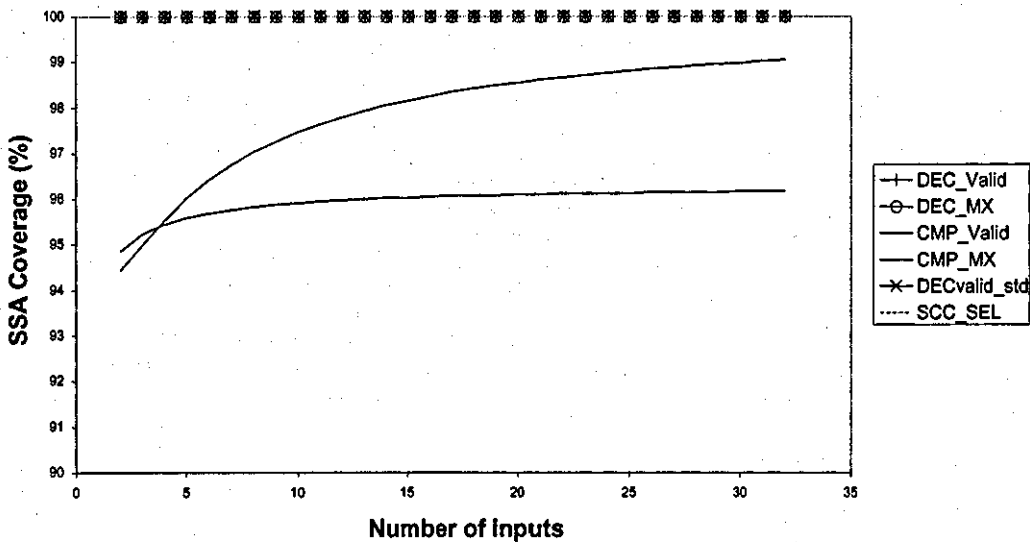


Figure 5-14 Maximum production test coverage.

**Maximum Protocol SSA coverage**

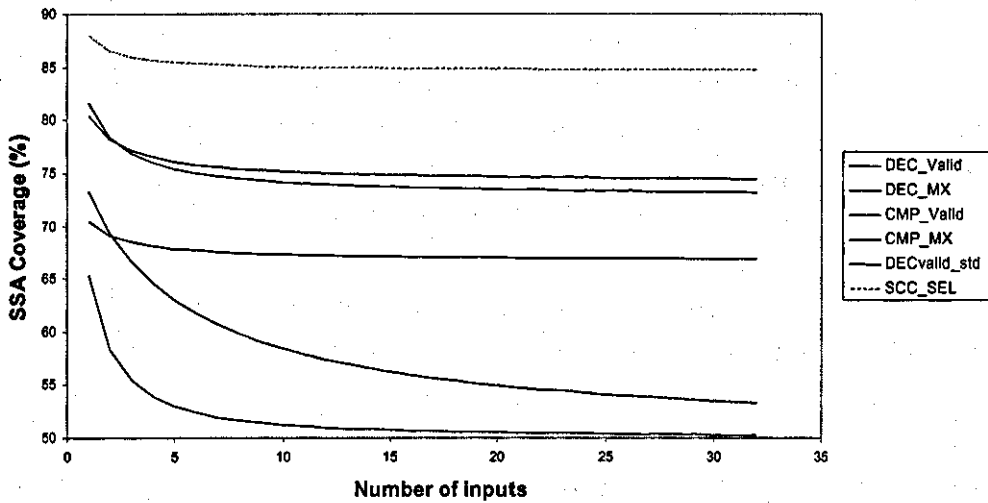


Figure 5-15 Maximum test coverage when restricted to protocol obeying vectors.

Figure 5-15 shows the effect of scaling on the maximum single stuck-at fault coverage provided by protocol obeying vectors. This is a comparison of the on-line testability exhibited by the various scaled architectures. The best on-line testability is provided by the self-checking checker based architecture (which has a dedicated selector design). The lowest protocol SSA coverage is exhibited by those transmission controllers implemented using the combined (non-partitioned) architecture.

**Maximum Protocol + Freeze SSA coverage**

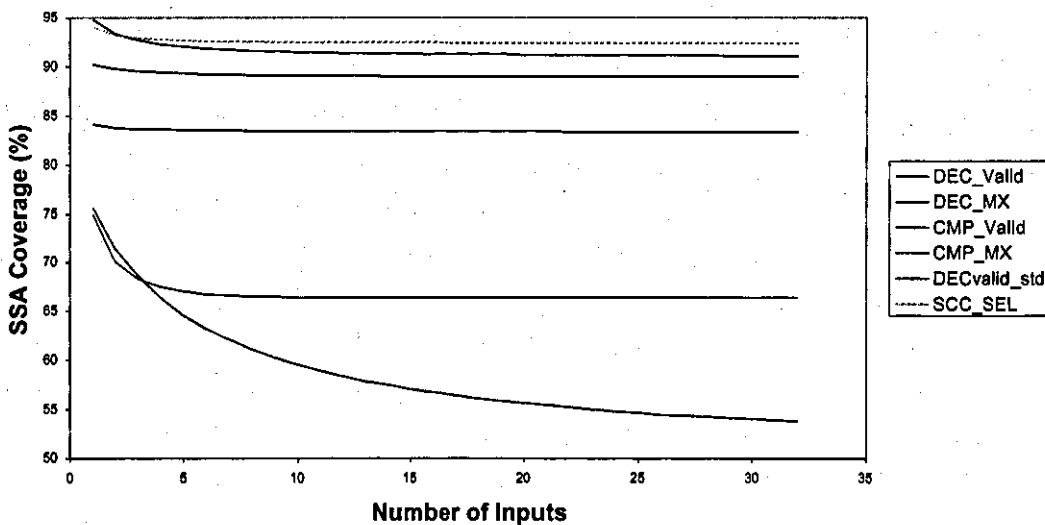


Figure 5-16 Maximum test coverage when restricted to protocol obeying vectors followed by an enforced freeze.

In addition to protocol obeying vectors, test coverage can be increased by initiating a system halt. This causes the IFIS cells within the system to freeze. Figure 5-16 shows the combined maximum single stuck-at fault coverage which results from maximum protocol potential followed by such a freeze. Figure 5-16 shows that the difference in SSA coverage between the architectures that employ partitioning, is less following a freeze that shown in Figure 5-15.

## 5.6 CONCLUSIONS

Comparison of four different techniques for designing the functional block within IFIS processing elements offers evidence that architectures based on duplication exhibit the highest on-line test coverage (100%) and the highest complexity. Additionally, this technique does not reduce the data throughput in comparison to conventional binary systems and does not restrict the range of applications that can be realised. This approach has the additional advantage over the other techniques in that it facilitates design construction due to its modular nature. Furthermore, it simplifies manufacturing test vector generation by permitting exploitation of vectors originally intended to test the conventional binary equivalent design. While the complexity overhead associated with this technique is unattractive, it is the only technique of those examined that fulfils the test coverage, application range and data throughput requirements specified earlier in this thesis.

The degree of checking that can be achieved without resorting to periodic off-line testing reflects the level of confidence that can be placed in an on-line testable system at any point in time. It is clearly advantageous for this confidence level to be high. Comparison of alternative techniques for achieving data flow control, based on incoming data validity, show that specific architectures based on comparators are not 100% testable even when unrestricted access is afforded to cell input/outputs. For an on-line testable system to display such attributes is clearly unacceptable.

Of those techniques that provide 100% production test coverage, the best compromise

between on-line test coverage and complexity is, perhaps surprisingly, not displayed by designs derived from a self-checking checker. This is because the selector circuitry required to utilise an SCC itself contains untestable nodes. Other techniques aimed at removing the single signal that controls data flow all result in significant complexity penalties in addition to an increase in the number of untestable nodes.

Based on the above discussion, it would seem that a favourable compromise between design complexity and on-line test coverage (without the need for a production tester) is provided by the simple decoder and conventional multiplexer approach. While this does leave the possibility of a single point of failure (select signal), this need not become a weakness if the remainder of the IFIS computation block is code disjoint. This is because a non-codeword appearing at the computation block (cell) inputs results in a non-codeword cell output in this case. This is then detected by the neighbouring IFIS cell, causing a system halt.



## **CHAPTER SIX**

### **IFIS FEASIBILITY STUDY**

#### **6.1 OBJECTIVES OF CHAPTER**

This investigation is the final investigation in a series that explores the influence on design characteristics imposed by the coding/protocol and structure adopted by the IFIS methodology. This investigation aims to:

- Report the application of IFIS to a real-world design implemented in hardware.
- Outline the design experiences associated with designing using IFIS.
- Demonstrate the operation of the circuit both in fault-free mode and with a specific failure.
- Compare the performance, complexity and fault coverage of the conventional and IFIS equivalent implementations. This empirically quantifies the characteristics of a combined system built upon the results of the previous two investigations.

#### **6.2 IFIS UART DESIGN AND IMPLEMENTATION**

To verify the methodology, a representative design task has been undertaken. The UART was considered to fulfill this role as it was both a reasonably complex design and also includes busses, state machines and registers, all of which are frequently used design elements. To obtain comparative data for designs implemented using the IFIS on-line test methodology, a UART has been realised in both conventional and IFIS forms in FPGA technology. A survey was performed and the features offered by commercially available UARTs were identified. The following generic functions were included in the UART design:

- 7 bit serial data with single start/stop bits
- 1 bit even parity checking
- Parity/Framing/Overrun error flag generation
- Provision for external serial port clock
- Transmitter/Receiver data buffers
- Interrupt generation

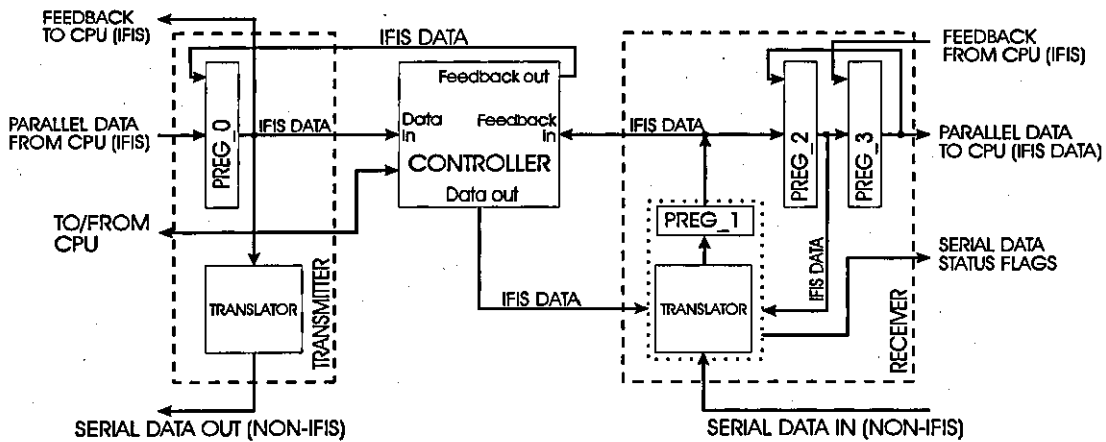


Figure 6-1 Top level architecture of the UART.

Figure 6-1 shows the control/data-flow graph of an IFIS UART. The UART shown in the figure and its behaviour were first defined by choosing from typical specifications of commercially available designs. Both versions of the UART were developed using this architecture. Figure 6-1 shows that the generic UART is partitioned into three blocks, namely the receiver, transmitter and controller. An outline of the function of each block now follows.

### 6.2.1 Receiver

The task of the receiver is to receive a serial bitstream in the form: start bits, data, parity information and stop bits and store the contained data until it is retrieved by the host CPU. The arrival of new data is signalled to the receiver by a transition from a sequence of stop bits to a start bit value at the serial data input. Following the reception of data, the receiver generates a set of error flags, intended for use by the

source of the bitstream. These flags reflect the validity of the data received by the receiver and may be used to trigger a re-transmission of the data if required. Simultaneously, the receiver communicates its status to the internal UART controller using asynchronous handshaking. This form of communication was required because the clock domain associated with the serial bitstream (and hence the receiver) is different from the clock domain associated with the host CPU (CPU clock domain) and controller. Received data words are stored in a First In First Out (FIFO) data buffer of depth 3. The receiver FIFO was implemented as 3 synchronous registers: each a self-contained IFIS cell in the IFIS UART. These are shown as PREG\_1, PREG\_2 and PREG\_3 in Figure 6-1.

### 6.2.2 Transmitter

The task of the transmitter is to take data words placed in its data buffer by the host CPU and convert them into a serial bitstream of standard format including start bits, data, parity information and stop bits. The parallel data lies within the CPU clock domain while the bitstream exists in the serial clock domain. The status of the transmitter is communicated to the internal UART controller using asynchronous handshaking. When no new data is placed into the transmitters data buffer it generates a bitstream at the serial data output containing only stop bits. PREG\_0 represents the transmitters data buffer in Figure 6-1.

### 6.2.3 Controller

This circuit controls data flow within the UART. This comprises two sub-tasks: Namely receiver FIFO management and transmitter FIFO management. Receiver FIFO management is to ensure that data is removed from the FIFO when the host CPU has read data from it and to inform the receiver that space exists in the FIFO. This occurs as a result of the CPU servicing an interrupt generated by the UART controller. Once the data has been removed from the receiver FIFO, the controller must also issue control signals that cause the data inside the FIFO to be moved. Thus allows more data to be added by the receiver and more data to be read by the CPU.

Transmitter FIFO management involves requesting new data from the host CPU when the FIFO is empty, generating the correct control signals to cause the FIFO to load data when supplied by the CPU, and informing the transmitter that data is in the FIFO. The controller issues different levels of interrupt requests to the host CPU depending on the data content of the receiver and transmitter FIFOs.

#### **6.2.4 Provision for IFIS**

In the IFIS UART, it was necessary to provide translation between IFIS codes and conventional external data and vice-versa because serial data arrives in a normal binary format. It was decided to place these translator units in the receiver and transmitter so that conventional data is transmitted/received via the serial ports. This allows the state machines controlling the Serial-In-Parallel-Out (SIPO) and Parallel-In-Serial-Out (PISO) registers to follow the same sequences of state transitions in both designs. These local state machines are implemented in a conventional manner. This is because no advantage is gained by implementing them in IFIS. Data is controlled within the UART by the controller which operates in parallel with the other state machines communicating with it using a handshaking protocol. If the controller halts due to a failure detected by the IFIS protocol, it is unable to generate an acknowledge signal. Since the local state machines do not receive the acknowledge signal they retain their current state and cannot direct data movement within the serial ports. Following the definition of block specification, the receiver, transmitter and controller blocks were designed independently following the IFIS structure selected in chapter 5 whenever IFIS cells were employed. The function blocks within each of the IFIS cells were designed such that single faults could only affect one bit of each (t,f) pair. This was necessary to ensure that the IFIS protocol would be disrupted following stimulation of a single fault.

#### **6.2.5 Design Issues**

The main issues associated with the using the IFIS methodology are design implementation and design verification. These issues are discussed below.

### 6.2.5.1 Design implementation

There are a number of ways in which IFIS designs can be implemented. The key to implementing IFIS designs is to exercise control over the structure of the netlist such that the computation block is of the generic structure shown in Figure 6-2.

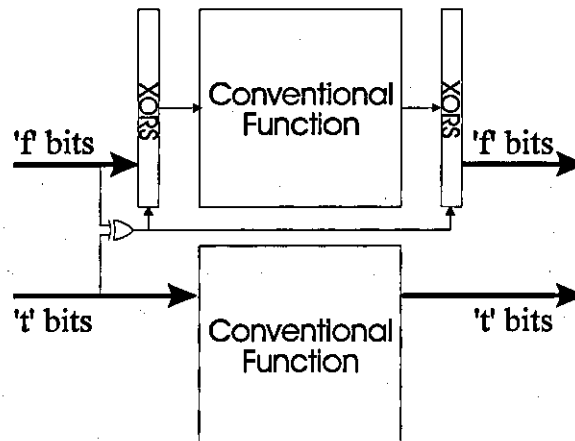


Figure 6-2 IFIS computation architecture.

Duplication is used because it is considered to be the most effective generic method for checking logical operations [Avizienis71a, Patel82]. This block structure is partly based on an idea that appeared in [Reynolds78] and [Shedletsky78] for the purposes of error detection and error masking respectively. This structure should be used because

- No single fault internal to the function can affect both rails of the dual-rail pair carrying an IFIS code. Active faults therefore always result in non-codeword output and violate the protocol.
- It is generic making it easy to understand and simple to implement. Both of these properties result in minimised design time and hence reduced cost.
- A direct structural relationship between the functionality contained in the conventional and IFIS implementations is established. This permits design structures of known characteristics to be used and facilitates test pattern generation for off-line testing.

The two primary factors that influence the final design structure are:

- Design description
- Design manipulation

When using an HDL in conjunction with the IFIS methodology, a key issue is the level of abstraction to use. A high level of abstraction is used to describe the behaviour of a design when its structure is unimportant. The fault detecting abilities of IFIS cells, as with most other design for test techniques, depend on the circuit structure. To obtain the structure for the function part of an IFIS cell, shown in Figure 6-2, requires that the conventional function be realised and then placed using a structural description. This technique must also be applied at the IFIS cell level if the cells are to be structured as described in chapter five. The style of description applied to the conventional function within the IFIS cell is left to the discretion of the designer for who additional control over the structure may not be required. It is, however suggested that it be identical to that used in the original conventional implementation so that advantage can be taken of existing test vectors/design verification.

Design manipulation occurs during optimisation. It can traditionally be applied at any level of the design hierarchy and results in the loss of hierarchical structure below the level to which it is applied. To maintain the structure of IFIS cells, a pre-requisite if the fault detection properties are to be maintained, the design manipulation should only be performed at the level of the conventional function block in both the conventional and IFIS implementations. The selector logic is made from paralleled standard cell multiplexors and thus optimisation brings no benefit. A similar reasoning applies to the storage section of the IFIS cell. The protocol checking logic can be optimised as long as the resultant structure is not an exclusive-or tree which for checking the IFIS protocol is not 100% testable. This is due to the internal fanout of the checker inputs.

### 6.2.5.2 Verification

Since there is a strong structural relationship between IFIS and conventional implementations, vector sets designed for functional verification (and structural testing) of both implementations should be strongly related. By ensuring that the same data sequences are applied to the primary inputs of both implementations, the same sequence of internal states occur in each under fault free conditions. The advantages of this are:

- Once confidence is gained in the correctness of a conventional design using a functional vector set, the investment already made in the vector set can be exploited when verifying the IFIS design.
- Vectors designed to test the structural integrity of the conventional implementation can be manipulated to test the structure of the corresponding IFIS implementation, thus reducing the costs of test vector generation for use during production test.

The UART designs were verified by applying input vectors to software models of the UARTs written in the 'C' programming language and storing the responses. The vector sets were related in the way described above. Figure 6-3 shows the design flow used to generate the stimulus files for the UARTS.

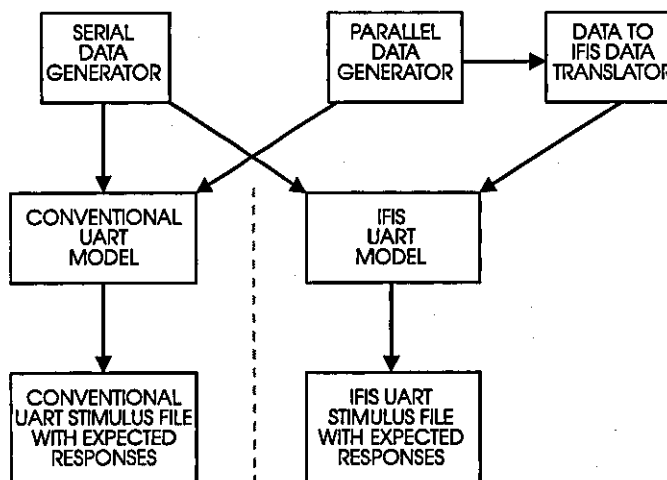
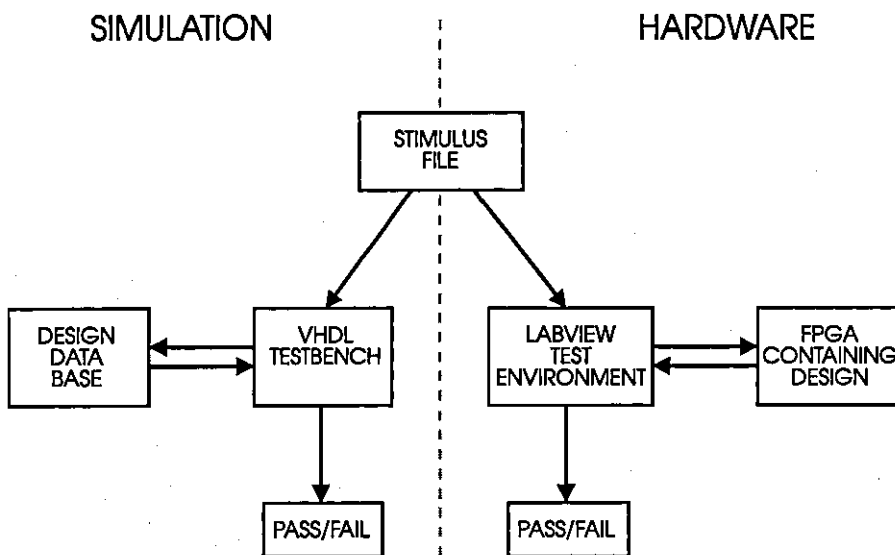


Figure 6-3 The flow used to generate verification patterns for the UARTs.

Examination of Figure 6-3 shows that serial data is applied to both the IFIS and to the conventional UARTs directly. This data is intended for application to the serial port of the UARTs. Serial data ports were not implemented using IFIS coding for the reasons described in section 6.2.4. However the data generated for application to the parallel data port (to/from the host CPU) requires translation prior to application to the IFIS UART because the parallel port is implemented using the IFIS protocol in the IFIS UART.

When simulating, vector files containing input vector sequences and expected outputs were read by VHDL testbenches and the simulation responses compared to the 'C' model responses. When the designs were placed in FPGAs, the Labview environment running on a PC was able to apply the same vector files as those applied to the simulations and use them to check the FPGA responses. Figure 6-4 shows the similarities between verification of the design data base and the verification of the hardware implementations.



*Figure 6-4 Design verification in the software and hardware environments.*

Using this method of test pattern application following the generation of a reference test response set, it was possible to generate and apply large vector sets which increased confidence in the similarity in behaviour between the 'C' based models and the VHDL sourced design.



### 6.2.5.3 Fault injection

To demonstrate a fault condition, a means for fault injection was added to the UART design. Figure 6-5 shows the position of the injected fault, marked by the symbol  $\alpha$ , within the controller block which itself comprises two IFIS cells. The insert to Figure 6-5 shows the logic for injecting the example fault under control of DATA\_FLT.

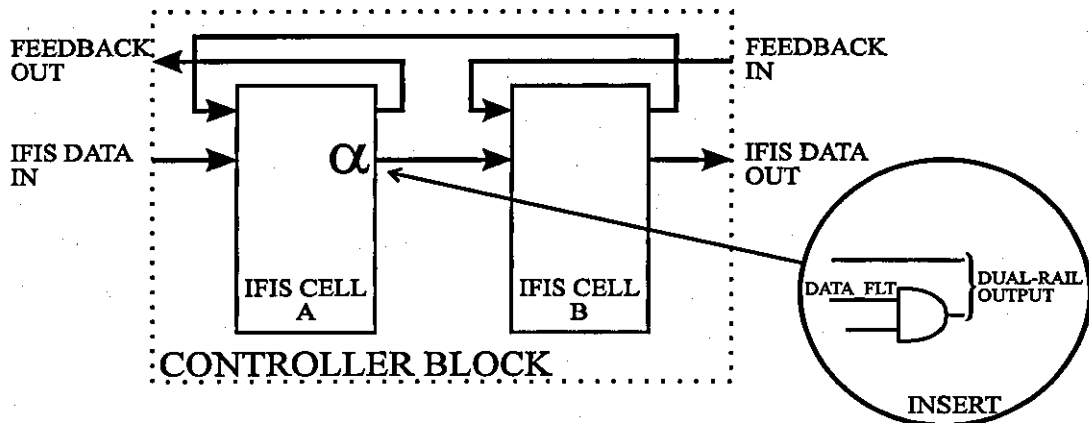


Figure 6-5 The injected fault, marked by  $\alpha$ , activated in the controller.

Since the error detection abilities of dual structures are well known, this fault was chosen to demonstrate the effect of an error occurring at an IFIS cell boundary. The equivalent of a stuck-at 0 fault can be applied to a single bit of the code generated at the output of the cell labelled IFIS cell A. Stimulation of this fault results in a non-monotonic transition between the last value and the current value held on the affected dual-rail pair.

## 6.3 RESULTS

This section contains the results obtained from implementing the UART using the two design methodologies. The design flow used was that from VHDL through synthesis and optimisation, then finally to Xilinx netlist. The Xilinx netlist was placed in the FPGA, using the Xilinx software provided.

### 6.3.1 Faulty and Fault-free Operation

Table 6-1 relates the signal names occurring in Figure 6-6 and Figure 6-7 to the interfacial descriptions shown on Figure 6-1 with the exception of the clock signals, reset request line and test mode select, which were omitted from the figure for reasons of clarity.

Signals	Signal Descriptions
CLK, S_CLOCK	Global and Serial clock signals
RRQ, TMS	Reset request and Test Mode Select
IACK, IRQ0, IRQ1	Controller - host CPU handshaking
PAR_ERR, FRAM_ERR, OVRN_ERR	Parity, Frame and Overrun errors
SDI, SDO	Serial Data In / Out
RX_T_BUS, RX_F_BUS	Parallel Data from UART to CPU (IFIS coded)
FB_T_IN, FB_F_IN	IFIS Feedback from CPU to UART
TX_T_BUS, TX_F_BUS	Parallel Data from CPU to UART (IFIS coded)
FB_T_OUT, FB_F_OUT	IFIS Feedback from UART to CPU

Table 6-1 Signal names and descriptions.

Figure 6-6 shows the FPGA containing the IFIS UART operating under normal conditions. Figure 6-7 shows what happens when a fault is introduced which affects the output of an IFIS cell, in this case a cell inside the controller.

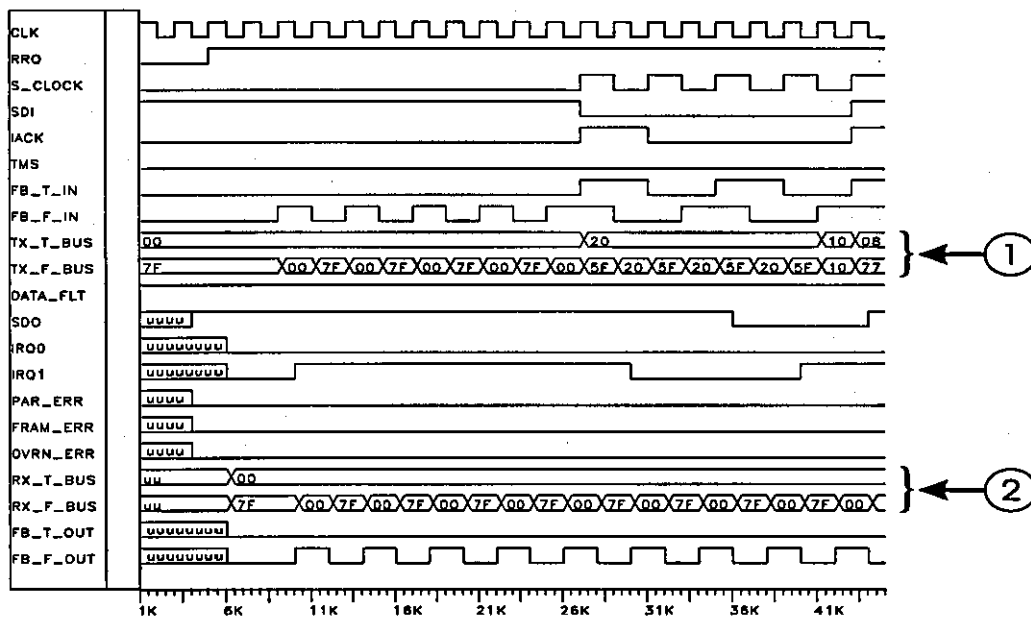


Figure 6-6 The IFIS UART under normal operation.

The arrows that appear on Figure 6-6 and Figure 6-7 denote the following:

1. The IFIS data bus from the CPU carrying IFIS coded data to the UART.
2. The IFIS data bus to the CPU carrying IFIS coded data from the UART.
3. The activation of the fault at a by signal DATA\_FLT.
4. The deactivation of the fault.

Examination of Figure 6-7 shows that following fault injection, and excitation, the RX\_T\_BUS and RX\_F\_BUS collectively halt. That means that both busses retain data. In this case, the fault was not excited by incoming data immediately following injection. This is evident by the delay in halting of the RX\_T\_BUS and RX\_F\_BUS outputs. Once halted, this status remains even after the removal of the fault.

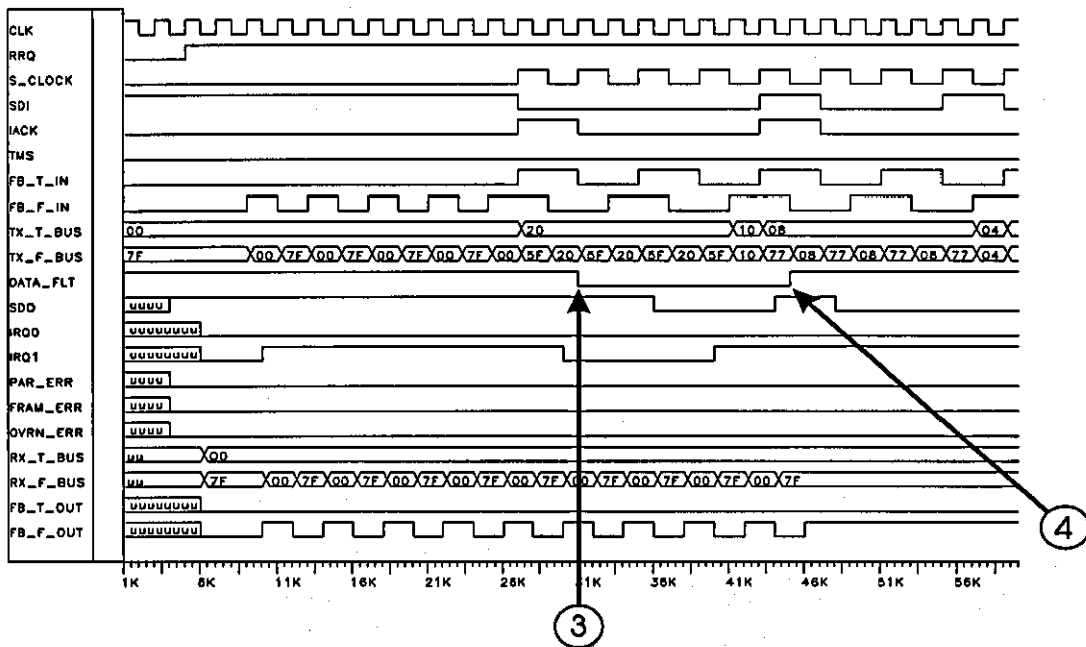


Figure 6-7 The IFIS UART under an injected fault condition.

Under normal conditions this halting would also trigger the CPU to no longer supply data via the TX\_T\_BUS and TX\_F\_BUS, however the worst case scenario occurs when this halting is ignored by the host CPU which continues to supply data. This is the case in Figure 6-7.

### 6.3.2 Complexity and Performance

To obtain comparative results for pure IFIS circuitry and conventional circuitry, the controller was implemented purely in IFIS. For completeness comparative results for all blocks with respect to complexity and performance have been included.

Table 6-2 shows the comparative complexities for the two implementations of the UART. The measures are of gate counts rather than configurable logic blocks (CLBs), which are the standard measure of complexity for Xilinx FPGAs, because the resolution is higher. The complexities were extracted using 'Logsyn' under the Intergraph v12.01.2 design suite.

Block Name	Implementation		Relative Complexity
	Conventional (Std. Cells)	IFIS (Std. Cells)	
Controller	135	301	2.23
Receiver	183	375	2.05
Transmitter	87	148	1.70

*Table 6-2 Complexities of the three top level UART design blocks.*

Examination of data relating to the controller in Table 6-2 reveals that the IFIS implementation is a 2.23 times as large as the conventional counterpart. This can be attributed to a combination of function duplication, as shown in Figure 6-2, and the overhead incurred by the necessity of having transmission control logic. The overhead related to the checker logic is directly proportional to the number of IFIS cell inputs, while the overhead associated to the selector is directly proportional to the number of IFIS cell outputs.

Block Name	Maximum Frequency (MHz) Conventional	Maximum Frequency (MHz) IFIS	Relative Performance
Controller	12.0	9.0	0.75
Receiver	9.4	9.4	1.00
Transmitter	18.5	16.1	0.87

*Table 6-3 Performance of the three top level UART design blocks.*

Table 6-3 shows the maximum frequencies of each of the design blocks according to the back-annotated netlist data-base. The target technology was a Xilinx 4013 FPGA (theoretical maximum frequency of 50MHz). It was not possible to verify these results by applying test vectors to the FPGA at such high speed due to limitations imposed by the PC based test bed. Since IFIS uses a saturated dual-rail code (as described in chapter four) and data changes during every clock cycle, its data rate is the same as the code rate which is 1data state/clock.

Examination of data relating to the controller in Table 6-3 reveals that the IFIS implementation is 0.75 as fast as the conventional counterpart. This can be attributed to the additional signal delay from conventional function outputs to storage inputs by the additional exclusive-or gate, shown in each F bit path in Figure 6-2, and the selector circuitry described in chapter five.

### 6.3.3 Fault Coverage

For the purpose of obtaining comparable fault coverage information, the 'C' models described in section 6.2.5.2 were developed to reflect the top level partitioning. Thus each UART model comprises models of the three major sub-blocks, each with interface lists matching their hardware counterparts. The interface lists of the conventional and IFIS controllers differ due to the coding used in the IFIS implementation. However, the control sequences issued by each controller under similar input conditions result in the same data flow in both UARTs.

Using the models it was possible to extract the data sequences occurring at the controller interfaces from identical top level data sequences applied to each of the two UART implementations. Standalone netlists of the two controllers were separately subjected to these local vector sequences and the cumulative fault coverage for each sequence calculated using the TDX fault simulator. In IFIS systems, any logical deviation reaching the IFIS cell boundary compromises the IFIS protocol thus triggering a system halt. For this reason, the fault coverage measured at the controller interface is representative of the on-line coverage which would be exhibited if the

UART were subjected to the same top level data sequence.

The best case and worst case fault coverage predictions for the IFIS design can be approximated from the cumulative coverage observed for the conventional implementation. This is because the structural relationship between the two implementations is known to be of the form shown in Figure 6-2. The minimum predicted coverage corresponds to the case where no cumulative fault coverage is expected within the transmission control logic shown in Figure 5-9 and Figure 6-2. The maximum predicted fault coverage would occur if the gates within the transmission control logic are 100% tested and contribute to the overall coverage.

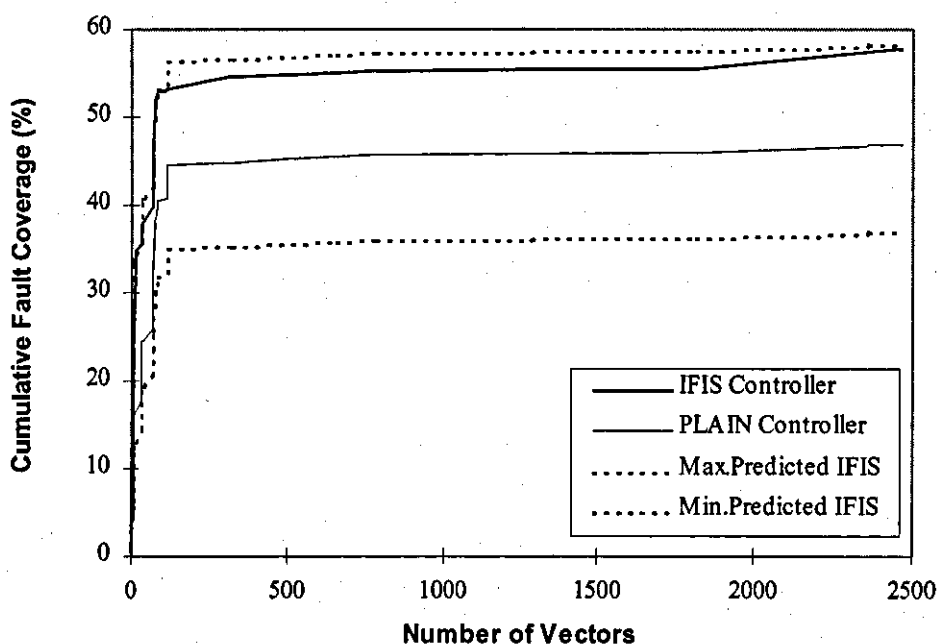


Figure 6-8 Comparison of SSA fault coverage.

Figure 6-8 shows the cumulative fault coverage exhibited by the conventional controller, the maximum (minimum) predicted coverage assuming 100% (0%) fault coverage of the additional circuitry in the IFIS implementation and the observed coverage in the IFIS implementation. It is important to note that it is the relative coverage observed in the IFIS system with respect to the predicted IFIS values (and not the absolute values) that characterises the testability of the IFIS control circuitry. Following the application of approximately 1800 vectors, a fault was injected, and the

observed fault coverage in the IFIS controller is seen to rise correspondingly.

Unlike off-line testing, the test time is infinite in the on-line test environment. However, comparison of the fault coverages shown in Figure 6-8 (which covers a finite period of time) shows that the increase in complexity incurred in the IFIS implementation is offset by the ability to provide on-line testing with a high fault detection ability. Furthermore, the additional complexity introduced by duplication and data flow control logic is shown to be well tested by vectors which were not specifically targeted for that purpose.

#### **6.4 SUMMARY**

The feasibility of the IFIS methodology has been explored using a contemporary IC design. A UART was chosen for implementation because it contains examples of features typical of state of the art designs, namely: Serial to parallel converters, parallel to serial converters, state machines and synchronous registers.

The UART was implemented on FPGA and its operation was verified using vectors generated by an independent functional model. In addition to functional verification, the design was characterised with respect to data throughput, complexity and behaviour under fault conditions. Additionally, a method to assess on-line test coverage was derived and applied to the design. This enabled the assessment of the on-line test coverage enjoyed by an IFIS state machine buried deep within the UART design.

#### **6.5 CONCLUSIONS**

From the results obtained, it can be concluded that the concept of the IFIS methodology is sound and additionally, that the halting mechanism provided by IFIS successfully provides error detection, error flagging and error management.

Furthermore, the combination of protocol/encoding scheme selected in chapter four with the cell structure derived in chapter five results in systems that:

- Display the same data throughput rate as their conventional binary equivalent systems while supporting unmodified data flow during error-free operation.
- Display a high level of on-line test coverage. Despite the increase in complexity, a substantial fraction of the additional hardware required for the implementation of IFIS designs is tested concurrently with the hardware that it protects.
- Provide the intended permanent halting behaviour under fault conditions.
- Do not permit erroneous data to propagate to primary system outputs.

As expected, the complexity of IFIS designs is more than double that of their equivalent binary implementation. Two factors affect the complexity of IFIS designs, namely:

- The duplication required for providing a dynamic functional reference. This is a direct consequence of incorporating enough flexibility into the methodology so as not to restrict the range of applications to which it may be applied.
- The overhead required for checking information communicated between IFIS processing elements. The simplicity of the dual-rail-coding scheme allows this dedicated hardware to be built in a modular way, thus resulting in a predictable complexity for checking a specific number of data bits.

Based on the above discussion, it can be concluded that to minimise the complexity penalty associated with IFIS, maximum functionality should be included within each IFIS processing element.



## **CHAPTER SEVEN**

### **FAILURE DIAGNOSIS IN IFIS**

#### **7.1 OBJECTIVES OF CHAPTER**

This investigation assesses the utility of knowing the sequence in which system outputs halt for performing failure diagnosis in IFIS systems. The objectives of this investigation are to:

- Develop a strategy for failure diagnosis in IFIS
- Identify the limitations of the strategy
- Describe the practical application of the diagnosis technique
- Demonstrate the application of the technique to hardware
- Empirically identify the practical limitations of the technique
- Explain the results obtained from the technique application

#### **7.2 MOTIVATION**

Failure diagnosis is useful to the IC manufacturer because its output can be used to identify faulty regions within designs. When a number of manufactured ICs fail in the same region this can often be attributed to manufacturing process problems. In some cases, it is possible to adjust the physical implementation of future versions of the same IC design to reduce the chance of problems occurring in the same region [Sherwani95].

Any mechanism that assists with failure diagnosis of repairable systems has the potential to increase system availability by reducing the time taken to locate a replaceable unit.

On-line testable designs inherently exhibit some of the properties that are needed to perform failure diagnosis. The common tasks of monitoring, modelling and partial analysis, can be identified by comparison of Table 7-1 and Table 7-2 that refer to representative diagnosis and on-line test techniques respectively.

	<b>Expert System [Bernard94]</b>	<b>'Golden' Device [Purcell88]</b>	<b>Dictionary [Aitken95]</b>	<b>Consistency checking [Girard95]</b>	<b>Elimination [Cox88]</b>
<b>Monitoring.</b>	Device outputs	Device outputs	Device outputs	Device outputs	Device outputs
<b>Modelling</b>	Model contained in rule base	Device used as reference	Model described in matrix	Individual component models	Component / connection models
<b>Analysis</b>	System traverses rule base	Differences highlighted	Quality of match is assessed	Localise inconstancy	Analyse history of node values
<b>Hypothesis</b>	Candidate list (probable causes)	Physical region	Candidate list (probable causes)	Candidate list (probable causes)	Candidate list (possible causes)

*Table 7-1 Classification of activities for different methods of fault diagnosis.*

In Table 7-1, the tasks of monitoring, modelling, analysis and hypothesis are mapped to the appropriate activity for specific diagnosis techniques. Table 7-2 maps the techniques of monitoring, modelling and analysis to tasks appropriate to on-line testing.

	<b>Information Redundancy</b>	<b>Hardware Redundancy</b>	<b>Time Redundancy</b>
<b>Monitoring</b>	Design block outputs	Design block outputs	Design Block outputs
<b>Modelling</b>	Model contained in code	Spare hardware used as reference	Previous behaviour used as reference
<b>Analysis</b>	Design block Pass/Fail	Design block Pass/Fail	Design block Pass/Fail

*Table 7-2 Classification of activities for different methods of on-line testing.*

The high proportion of activities common to both on-line testing and to failure diagnosis suggest that on-line testable systems can themselves assist in system level

diagnosis, thus reducing the need for external intervention. IFIS systems must therefore also be capable of assisting in the diagnosis of system failure.

The halting property associated with IFIS designs distinguishes them from the other on-line test techniques. IFIS inherently performs error management following error detection. The halting mechanism that provides this feature is triggered following error detection at an IFIS cell input. The cell that detects an error condition ceases processing, thus causing an error condition to appear on monitored inputs of immediately neighbouring cells. In this way, halting of IFIS cells progresses throughout the entire IFIS system, progressing away from the error source at the rate of one cell per system clock cycle until all primary system inputs and outputs are reached. The way in which IFIS systems halt following error detection suggests that diagnostic information be contained in the halting sequence of system outputs. This investigation assesses the ability to perform failure diagnosis within IFIS designs based on the halting sequence of IFIS system outputs.

### **7.3 EXPECTATIONS OF FAILURE DIAGNOSIS IN IFIS**

While the inherent error management characteristic of IFIS systems results in diagnostic information, it also affects the range of diagnostic techniques that can be applied to them.

#### **7.3.1 Inherent Influences of IFIS**

The effects of preventing erroneous data from being processed/propagated are associated with a number of advantages and disadvantages in the context of failure diagnosis. The advantages associated with this mechanism are:

- The activities of monitoring, modelling and analysis reduce the workload on external failure diagnosis equipment. This facilitates failure diagnosis in the manufacturing environment and can be exploited to provide in-the-field

diagnostic capability with minimal external intervention. This must result in increased system availability.

- There is no need to re-stimulate a known faulty design with tests that are designed to test for specific fault conditions.
- The information required to perform diagnosis is simplified by referencing information rather than using actual data values. Information de-referencing was used to perform diagnostic simplification in [Aitken95].
- Failure diagnosis of state machines using explicit post test diagnosis is feasible in IFIS because of the abstract nature of the available information. Explicit post test diagnosis is commonly restricted to combinational designs [Maly91, Millman91, Somayula93, Chang70, Tsiang62].

The disadvantages associated with the halting mechanism are:

- It makes IFIS designs unsuitable candidates for multiple defect diagnosis. Multiple defect diagnosis can be performed on conventional binary designs by using implicit methods based on analysing results obtained from the application of predetermined test vector sets (Post test diagnosis) [Aitken91, Aitken92, Millman94, Waikuk89]. Additionally, implicit diagnosis can be performed on conventional designs using interactive test vector selection (Diagnostic testing) [Cox88, Abromov80]. In both cases, multiple examples of failure must occur before diagnosis is considered to be achieved. IFIS systems stop on the first occurrence of error (SOFE).
- When diagnosing IFIS systems, the resolution achievable for single defect diagnosis cannot be guaranteed when using explicit diagnostic testing because of the SOFE characteristic of IFIS. Explicit diagnostic testing requires less storage capacity than that required for post test diagnosis [Girard95, Bernard94, Brugnioni93, Favila91, Purcell88, Genes84, Kautz68, Chang65]. For this reason this class of techniques is considered preferable for performing single defect diagnosis in conventional designs.

The listed disadvantages are generally associated with defect analysis that results in

information useful for adjusting the manufacturing process. This level of resolution is not necessary for performing diagnosis within repairable systems.

### 7.3.2 Expected Limitations on Resolution Based on Halting Sequence

The purposes of identifying the limitations associated with failure diagnosis based on the halting sequence of IFIS system outputs are twofold, namely:

- To discover the potential usefulness of the technique.
- To identify design restrictions that may be required to maximise the abilities of the diagnosis technique.

An informal discussion of the expected limitations in diagnostic resolution based on the halting sequence of IFIS primary outputs is presented below.

The protocol adopted for IFIS designs was selected in chapter four. The selected protocol is inelastic and halts permanently following fault detection. The effect of the outputs of a specific cell signalling an error are to initiate a halt which propagates throughout an IFIS system at the rate of one cell per clock cycle. Halting is bi-directional. This means that IFIS cells connected to primary inputs, in addition to those connected to primary outputs, cease to process when the protocol is no longer satisfied. This means that once a cell initiates a halt a complete system halt follows.

The most complex diagnosis tasks are those where the minimum diagnostic information is available. To satisfy the IFIS protocol the system feedback output (coming from a primary input cell) must be monitored in addition to the output of the primary output cell. In this way IFIS systems provide access to a minimum of two system outputs.

It can be inferred that a path must exist from each primary input to each primary output via any internal cell. If this were not the case certain cells would exist for which local error detection would not cause a complete system halt. When

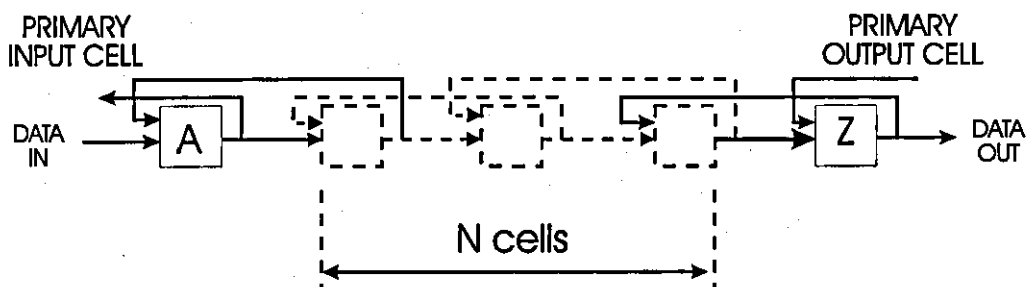
considering the number of paths existing between an IFIS primary input cell and an IFIS primary output cell, two scenarios exist, namely:

- There is only one path
- There are multiple paths

### 7.3.2.1 Single path scenario

Figure 7-1 shows a generic IFIS chain (as presented in chapter four) where only one path exists between a primary input cell and a primary output cell which are separated by  $N$  cells.

The structure of the design influences the time taken (in clock cycles) for data to progress through the design. Therefore, the position of each cell in the data path is a fixed number of cells (clock cycles) from both the primary input cell and the primary output cell. Thus the time taken (path length, measured in clock cycles) for an initiated halt to become apparent at the primary input and primary output cells is unique for each initiating cell.



*Figure 7-1 An IFIS chain where only one halting path exists from the primary input cell to the primary output cell.*

When monitoring primary system outputs, the absolute time taken from internal error detection to primary output halting is not available. The available information is the difference in time taken between the halting of the first primary output and the halting of the others. In the scenario shown in Figure 7-1, all cells can be distinguished using the reduced information.

### 7.3.2.2 Multiple path scenario

Where multiple paths exist between two cells a minimum of one closed loop exists between the input cell and the output cell. Once a loop exists, it is not possible to distinguish between all cells in the loop if only two cell outputs are monitored.

Where one closed loop exists, two halting paths exist from each cell to each primary output. Figure 7-2 shows a simple generic multiple-path IFIS system. In this system, a clockwise path exists from each cell to both the primary input cell and to the primary output cell. Similarly, an anti-clockwise path exists from each cell to both the primary input cell and to the primary output cell. This is due to the bi-directional communication used in IFIS systems.

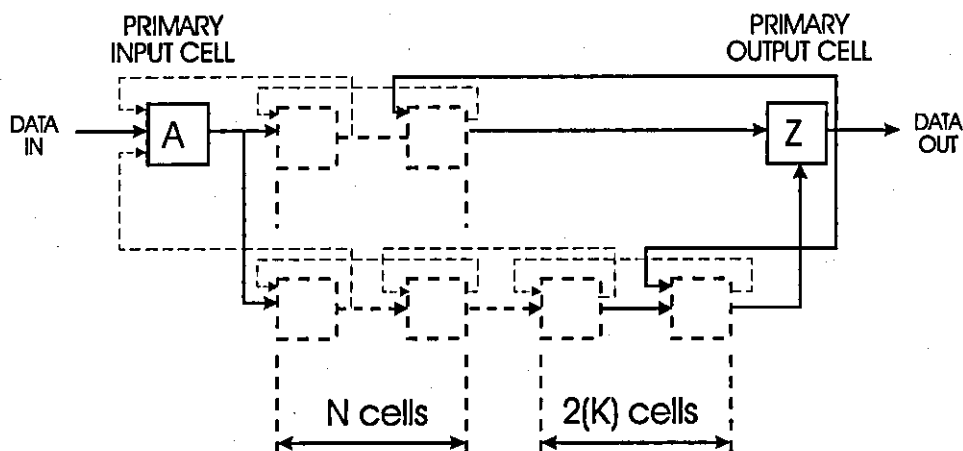


Figure 7-2 Generic IFIS system where two paths exist between cell A and cell Z.

The inelastic nature of the protocol dictates that the length of all paths from the primary input cell to the primary output cell must either be identical or differ by an even number of cells. Referring to Figure 7-2, K can be zero or any integer. The ability to distinguish between cells is described using examples.

#### Example 1: $K=0$

Figure 7-3 shows an example IFIS system of the type shown in Figure 7-2 where  $N=3$  and  $K=0$ . In this example, the path lengths from cell A to cell Z are equal in length. For clarity, the IFIS feedback and data have been combined and denoted as bi-

directional arrows on the figure. Cells are shown as circles.

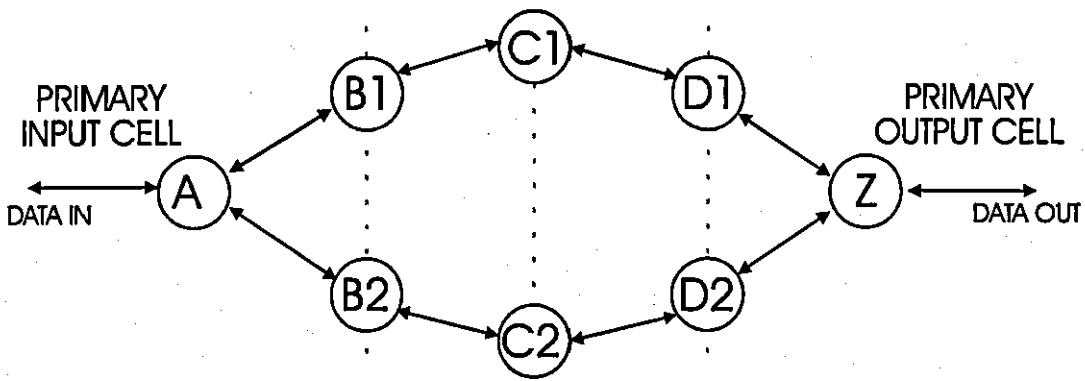


Figure 7-3 An example IFIS system where two paths of equal length exist between the primary input cell and the primary output cell.

While cell B1 is distinguishable from cell C1 and cell D1, it is not distinguishable from cell B2. Similarly, cell C1 is not distinguishable from cell C2. This is due to the symmetry exhibited in the paths.

Example 2:  $K=2$

Figure 7-4 shows an example IFIS system where  $N=3$  and  $K=2$  using the same notation as in Figure 7-3. In this example the multiple path lengths from the primary input cell, A to the primary output cell, I, are different depending on the route taken.

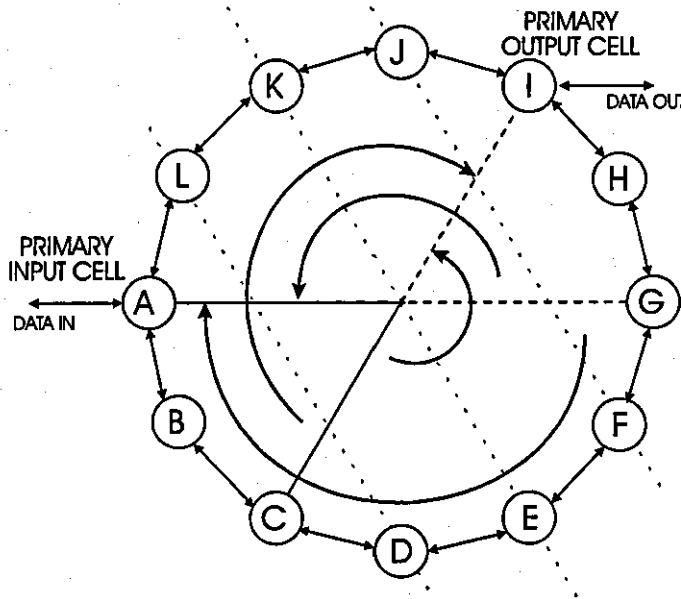


Figure 7-4 An example IFIS system where two paths of unequal length exist between the primary input cell and the primary output cell.



It was stated above that each path length from the primary input cell to the primary output cell (such as that shown in Figure 7-4) is identical or the path lengths differ by an even number of cells. This implies that the loop length, given by the number of cells contained in the closed loop, is always even. This further implies that for any monitored cell in the loop (either the primary input cell or the primary output cell) there exists only one cell at the maximum path length from it. In Figure 7-4 cell 'G' is the furthest away from cell 'A' and cell 'C' is the farthest cell from cell 'I'.

Halts initiated at cells C, B, A, L, K and J propagate via the clockwise path to cell I. Similarly, halts initiated by cells C, D, E, F, G and H propagate via the anti-clockwise path to cell I. This is because the IFIS protocol ensures that cell halting propagates via the fastest (shortest) path. A similar observation can be made for the halt propagation paths to cell A. Further examination of Figure 7-4 shows that when cells C, B and A initiate a system halt, it is propagated to both the output of the primary input cell and to the primary output cell in the same (clockwise) direction. This makes these cells indistinguishable as the initiator of a system halt. The reason that these cells are indistinguishable is that it is the difference between the halting times of system outputs which is observable. Fault diagnosis is based on this information alone.

CELL INITIATING HALT	ABSOLUTE HALT TIME OF CELL A	ABSOLUTE HALT TIME OF CELL I	OBSERVED HALT TIME OF CELL A	OBSERVED HALT TIME OF CELL I
A	1	5	1	5
B	2	6	1	5
C	3	7	1	5
D	4	6	1	3
E	5	5	1	1
F	6	4	3	1
G	7	3	5	1
H	6	2	5	1
I	5	1	5	1
J	4	2	3	1
K	3	3	1	1
L	2	4	1	3

*Table 7-3 Observable primary halting sequence of the system shown in Figure 7-4 mapped to the cells which initiate each halt.*

Table 7-3 shows the halting behaviour of the primary system outputs when halts are initiated at the cells shown in Figure 7-4. Examination of Table 7-3 further reveals that cell A and cell E are indistinguishable. Similarly, cells L and J are indistinguishable from cells D and F respectively. This is due to symmetry which was also apparent in the example with  $K=0$ .

### 7.3.3 Summary of Expectations

This section has demonstrated that the error detection/flagging mechanism used in IFIS designs can be used as a basis for fault diagnosis. Furthermore, resolution restrictions resulting directly from system architecture have been discussed using simple examples. System architecture was shown to influence the resolution attainable. Where only a single path exists from input cells to output cells resolution to the single cell can be achieved. Where multiple paths exist, this is not always possible. To predict the resolution restriction imposed by a typical system architecture requires knowledge of what represents a typical system. A first impression is obtained by applying the technique to the IFIS UART that was discussed in chapter six.

## 7.4 APPLICATION TO IFIS

In current designs implemented on ICs, the trend has been an increase in the ratio of the physical areas occupied by interconnect to the area of functional circuitry in standard cell designs [Sherwani95]. For this reason, it is inappropriate to ignore the effects of defects which affect interconnections. It is appropriate to determine if logic affecting faults (those which cause IFIS designs to halt) can be correctly diagnosed using the halting sequence apparent at the primary system outputs. This section therefore explores the effects of connection faults in addition to empirically determining the diagnostic resolution obtainable within the IFIS equivalent of a commercial UART.

The IFIS UART described in chapter six is used as a means to determine the ability to

perform diagnosis in IFIS systems under functional fault and connectivity fault conditions. The faults are induced and fault diagnosis is performed using an algorithmically generated dictionary of the type suggested in section 7.3.2.

#### 7.4.1 Fault Locations within the IFIS Structure

There are a number of general fault positions within the IFIS structure that affect the protocol in the event of occurrence. These are shown by  $X_n$  in Figure 7-5 (which reflects the IFIS cell design technique used in the IFIS UART) where  $n$  is an integer.

It should be noted that while the physical structure of an IFIS cell may result in the cell self feedback being downstream of the external feedback, this is unlikely because the self-feedback is internal to the cell. This means that there is no need to provide external access to the signal and therefore the shortest route from cell output to cell input can be used during cell design [Weste85]. This is not the case for the external feedback signals.

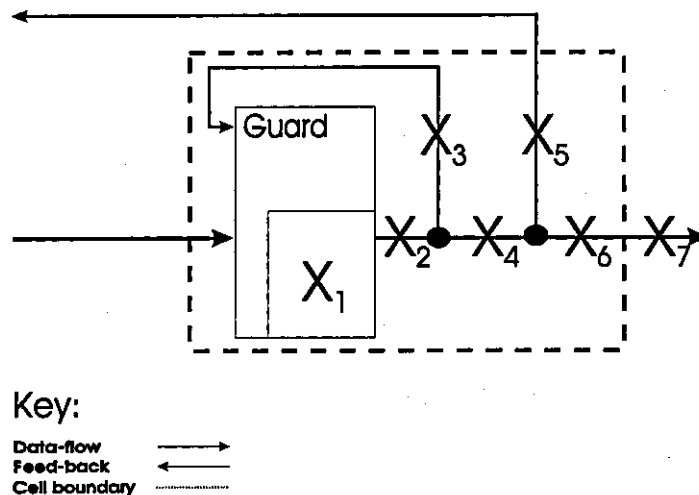


Figure 7-5 Protocol affecting fault sites.

$X_1$  represents a fault within the cell (not in the checker logic) which affects the cell output and is equivalent to  $X_2$ . Furthermore, faults  $X_1$  and  $X_4$  are equivalent in terms of halting sequence as long as the first halt shown at the cell output is considered in a single cell system. In a multi-cell system the system will remain frozen when a fault occurs even in the case of a single fault at  $X_4$ .  $X_2$  affects cell outputs and, by

implication, the feedbacks.  $X_3$  represents a fault occurring only on the self-feedback to the cell.  $X_4$  represents a fault that is equivalent to a combination of  $X_5$  and  $X_6$ . It must be noted that in all cases except that of  $X_7$ , cell outputs are affected with this model.  $X_7$ , however affects data to downstream elements without affecting the cell output itself. Based on this discussion, the unique fault types warranting examination are represented by those of type  $X_1$ ,  $X_3$ ,  $X_5$ ,  $X_6$  and  $X_7$ . Having reduced the fault list, it remains to identify relationships between the fault thus enabling the use of a simple algorithm to generate the dictionary.

**7.4.2 The Influence of Different Connection Faults on the Cell Halting Sequence**

The halting sequences presented in Table 7-3 assumed that the fault occurred upstream of the external cell feedback. Following the discussion presented in the previous section, this applies the model for an  $X_1$  fault. Figure 7-6 shows the interconnection of three neighbouring IFIS cells with the fault positions associated with the current cell superimposed.

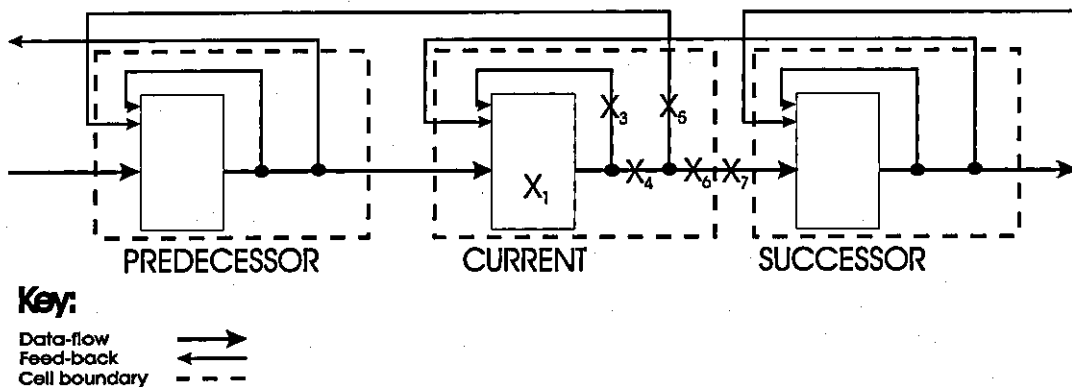


Figure 7-6 The context of each type of fault.

Table 7-4 describes the effects of named faults occurring at the current cell with respect to the halting times based on an  $X_1$  fault.

FAULT TYPE	INITIATION TIME FOR X1 MODEL		
	Predecessor Cell	Current Cell	Successor Cell
X1	-	1	-
X3	-	2	-
X5	2	-	-
X6	-	-	2
X7	-	-	2

*Table 7-4 Application of the X1 model to describe different faults.*

An explanation of Table 7-4 now follows. Assume that time is incremented at the beginning of each cell processing cycle. A fault is considered to occur during the first cycle, hence an  $X_1$  fault in the current cell occurs time 1. An  $X_3$  fault associated with the current cell directly affects the current cell. That is it causes the current cell to halt at time 2. Thus, the halting sequence associated with an  $X_3$  fault is the same as if it were an  $X_1$  fault delayed by 1 processing cycle. Similarly, an  $X_5$  fault associated with the current cell only directly affects the predecessor cell, which at time 2 halts. This is the equivalent of an  $X_1$  fault being sourced from the predecessor cell at time 2. Since all faults influence the halting sequence in a similar manner to that of the  $X_1$  fault with a modifier, only a description of the base algorithm is necessary.

### 7.4.3 The Halting Dictionary Generation Algorithm

The halting dictionary itself has already been defined to contain a complete set of halting sequences for all IFIS cells according to the models associated with  $X_1$ ,  $X_3$ ,  $X_5$ ,  $X_6$  and  $X_7$  faults. With this in mind any alternative algorithms that may be compared must result in an identical dictionary output, and hence must have no influence on the resolution obtainable for fault diagnosis. The remaining commonly used metrics for comparison are those of CPU time and memory requirement. More appropriate to my algorithms are a comparison of the number of node assignments made and the memory requirement. Two algorithms for dictionary generation are informally discussed and the pseudo code for the better algorithm is presented.

*Node Oriented Generation:* The dictionary is constructed by examining the effects of a halt occurring on the outputs of each IFIS cell within the system in turn. To achieve

this, each IFIS cell is in turn considered as the halt source and the halting times of the observable system outputs are calculated and stored. If  $n$  is the number of IFIS nodes in the complete system, then a minimum of  $n^2$  assignments are required to build the complete dictionary.

*Output Oriented Generation:* The dictionary is constructed by calculating the time taken for the outputs of each IFIS cell in the system to affect each of the primary system outputs in turn. To achieve this, each primary system output is considered in turn, storing the halt propagation times associated with the outputs of each IFIS cell. If  $n$  is the number of IFIS nodes in the complete system and  $p$  the number of dual-rail primary system outputs, then a minimum of  $p.n$  assignments are required to build the complete dictionary.

Both of the above algorithms are performed using models of the IFIS system and involve bi-directional path tracing. This may result in multiple attempted assignments associated with the outputs of each IFIS cell during calculation of each entry in the dictionary. It is for this reason that the actual number of assignments required to build the dictionary exceeds the minimum values provided above. To demonstrate the difference in efficiency of the two algorithms, each was applied to the same representation of the IFIS UART. The IFIS UART comprises 6 instances, 8 nets, 2 primary inputs and 2 primary outputs. Table 7-5 summarises the requirements for dictionary compilation for both algorithms when performed on the IFIS UART.

	<b>Node Oriented</b>	<b>Output Oriented</b>
<b>Model size (bytes)</b>	4120	4120
<b>Temporary storage (bytes)</b>	1032	1032
<b>N<sup>o</sup>. Of Node Assignments</b>	82	20

*Table 7-5 Characteristics of application of the different algorithms to the IFIS UART design.*

In Table 7-5, the model size is the quantity allocated to storing the design structure within the computer memory. It is calculated at the time of reading the design data base. The data structures used for performing each algorithm were identical. The temporary storage is the quantity of memory needed to store temporary variables

during the calculation of the dictionary. Table 7-5 shows that the only difference between the two algorithms are the number of node assignments. In this case, more than 300% penalty is incurred by the node-oriented algorithm with no offsetting advantage. For completeness, the pseudo code for the Output oriented algorithm is provided.

### Pseudo code:

```

Read netlist;
for each primary system output
node {
    set event time = '1';
    create two empty lists A and B;
    add the current node name to the list called A;
label: while A is
    not empty {
        for each node
        listed in A {
            for each cell
            connected to
            the node {
                for each cell
                output node {
                    associate the minimum value of 'event time + 1'
                    and it's already associated value of event time
                    with the node and then add it's name to B;
                }
            }
        }
        remove the node name from list A;
    }
    increment event time;
    if B is not empty {
        swap list labels A & B and goto label;
    }
    if B is empty {
        store the event values associated with IFIS nodes
        for this primary output in the halt dictionary;
    }
}
write out the halt-dictionary table.

```

### 7.5 FAULT INJECTION IN THE IFIS UART

The IFIS UART was used as a vehicle on which to perform fault diagnosis using real hardware. Figure 7-7 shows the IFIS view of the IFIS UART previously described in chapter 6.

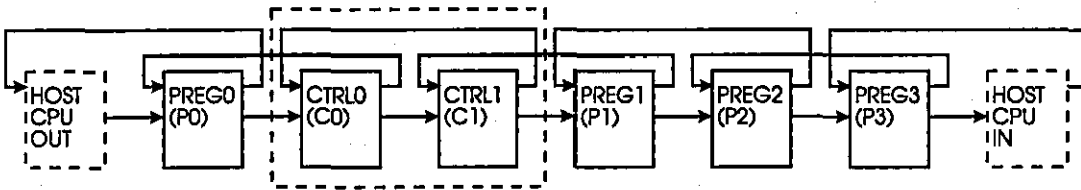


Figure 7-7 The IFIS structure of the IFIS UART.

For reasons of clarity the names in parentheses are those which are subsequently used to refer to the cells shown in Figure 7-7. The view shown in Figure 7-7 is that which is used by the algorithms described in section 7.4.3 to model the effects of faults within the system.

The host CPU cells are not part of the UART design but have been shown in the figure because the host CPU must provide/accept signals conforming to the IFIS protocol to/from the UART. The IFIS cells containing the prefix P form the FIFOs while the IFIS cells containing the prefix C form the UART controller which oversees all data movement to/from and within the IFIS UART. The controller is the most complex section of the UART, and is the most distant from the IFIS primary inputs/outputs. For these reasons, the CTRL1 IFIS cell, one of the 2 IFIS cells of the controller, was targeted for fault injection.

Faults are injected in the IFIS UART (specifically the C1 IFIS cell) via four signal pins, namely `function_fault`, `feedback_fault`, `int_feedback_fault`, and `data_fault`. Table 7-6 shows the mapping between signal pin names on the UART and the associated fault locations that are affected with respect to the selected controller block.



PIN NAME	FAULT
Function_fault	X <sub>1</sub>
Int_feedback_fault	X <sub>3</sub>
Feedback_fault	X <sub>5</sub>
Data_fault	X <sub>6</sub>

Table 7-6 Mapping of signal pin names to faults as shown in Figure 7-5.

To enable the pins to be referred to collectively, let us define a fault code to mean a four bit word made from the bits applied to activate/disable the four pins controlling locations X<sub>6</sub>, X<sub>5</sub>, X<sub>3</sub> and X<sub>1</sub> (left to right).

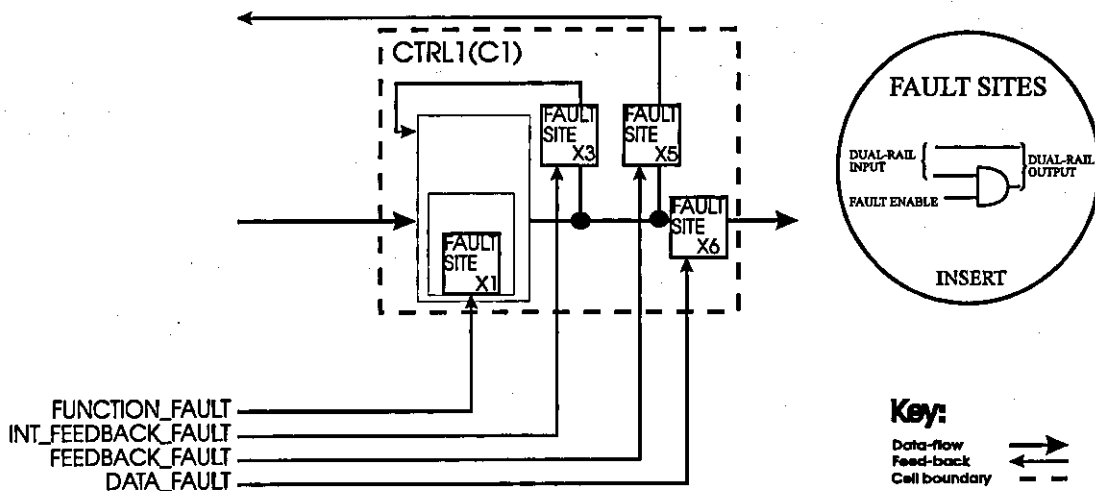


Figure 7-8 Fault injection sites within the selected IFIS cell contained in the IFIS UART.

Each of the named faults relates to the injection of a nominal stuck@zero fault affecting the data carrying bit of the IFIS bit pair. Figure 7-8 shows how the C1 cell has been adapted to permit fault injection within the controller. Faults are injected to the C1 block by activating signals corresponding to appropriate faults.

The fault injection signals are active low as shown in Figure 7-8. Advantage was taken of the multiple activation sites shown in Figure 7-8 to allow the UART to be subjected to fault combinations which are equivalent to other faults or for which no entry has been calculated in the dictionary.

Table 7-7 shows fault injection codes and the fault description which applies to each code.

Fault Code	Description
1111	No fault
1110	Function fault
1101	Internal feedback fault
1100	X <sub>1</sub> dominates feedback fault - function fault
1011	External feedback fault
1010	X <sub>1</sub> dominates feedback fault - function fault
1001	Multiple feedback fault
1000	X <sub>1</sub> dominates feedback fault - function fault
0111	Data Output fault
0110	X <sub>1</sub> dominates feedback fault - function fault
0101	Multiple fault, external feedback not affected
0100	X <sub>1</sub> dominates feedback fault - function fault
0011	Equivalent to X <sub>4</sub> fault
0010	X <sub>1</sub> dominates feedback fault - function fault
0001	Equivalent to an X <sub>1</sub> function fault
0000	X <sub>1</sub> dominates feedback fault - function fault

Table 7-7 The fault code placed on the fault injection channels and the code description.

Table 7-7 shows that by applying the code 0011, an X<sub>6</sub> and an X<sub>5</sub> fault, the equivalent of an X<sub>4</sub> fault is applied. Furthermore, it is possible to provide combinations of faults that have no equivalent (1001). The appropriate fault injection codes were applied to the IFIS UART during otherwise normal operation. The codes were later removed to discover the permanence of each injected fault. Before injecting different faults by changing the fault injection code, a system reset was performed and normal operation resumed.

## 7.6 RESULTS

Table 7-8 shows the results of performing diagnosis on the IFIS UART using a dictionary containing halting sequences for all possible X<sub>1</sub>, X<sub>3</sub>, X<sub>5</sub>, X<sub>6</sub> and X<sub>7</sub> faults on all IFIS cells within the design. The columns labelled actual fault and diagnosed fault list contain fault descriptions. The descriptions are of the format *Cell name: Fault type*. Therefore the fault code 1110 injects an X<sub>1</sub> fault in cell C1. The observed halting sequences at the primary outputs of the FPGA containing the IFIS UART were diagnosed as being consistent with a number of different possible fault sites. These

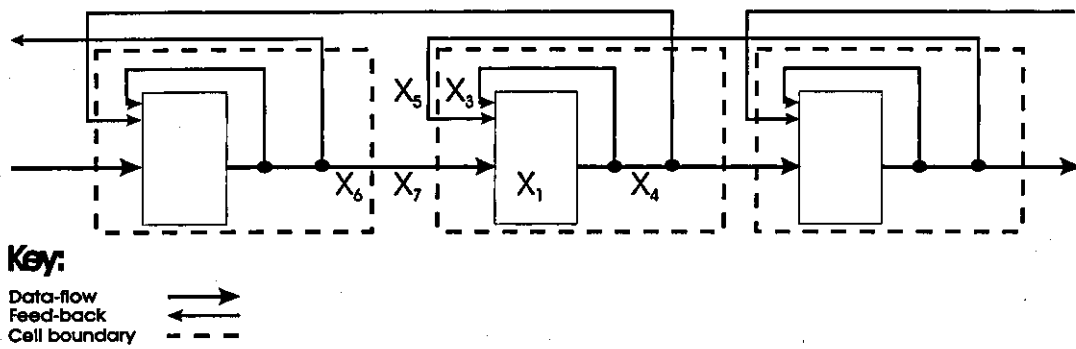
included an X1 fault in cell C1 and a number of other candidates.

<b>Fault Code</b>	<b>Actual Fault</b>	<b>Diagnosed Fault List</b>
1111	No fault	No fault
1110	C1:X1	C1:X1, C1:X3, P1:X5, C0:X6, C0:X7
1101	C1:X3	C1:X1, C1:X3, P1:X5, C0:X6, C0:X7
1100	C1:X1 (dominant)	C1:X1, C1:X3, P1:X5, C0:X6, C0:X7
1011	C1:X5	C0:X1, C0:X3, C1:X5, UFB:X6, UFB:X7
1010	C1:X1 (dominant)	C1:X1, C1:X3, P1:X5, C0:X6, C0:X7
1001	C1:X3, C1:X5	Not diagnosable (no match)
1000	C1:X1 (dominant)	C1:X1, C1:X3, P1:X5, C0:X6, C0:X7
0111	C1:X6	P1:X1, P1:X3, P2:X5, C1:X6, C1:X7
0110	C1:X1 (dominant)	C1:X1, C1:X3, P1:X5, C0:X6, C0:X7
0101	C1:X3, C1:X6	Not diagnosable (no match)
0100	C1:X1 (dominant)	C1:X1, C1:X3, P1:X5, C0:X6, C0:X7
0011	C1:X4 (X1)	C1:X1, C1:X3, P1:X5, C0:X6, C0:X7
0010	C1:X1 (dominant)	C1:X1, C1:X3, P1:X5, C0:X6, C0:X7
0001	C1:X1 (equivalent)	C1:X1, C1:X3, P1:X5, C0:X6, C0:X7
0000	C1:X1 (dominant)	C1:X1, C1:X3, P1:X5, C0:X6, C0:X7

*Table 7-8 Diagnosis of the injected faults.*

The data contained in Table 7-8 shows that diagnosed fault lists contain a specific grouping of fault types that are indistinguishable when halting of primary outputs is used for diagnosis. Closer examination of Table 7-8 reveals the following:

- Each diagnosed fault list contains only a single representative of each type of fault. Therefore if the type of fault were known, the parent cell can be identified.
- There is no situation that results in the false diagnosis of a non-existent fault. The results of diagnosis are therefore not misleading.
- In all cases the intended fault, the dominant fault or an equivalent fault was correctly identified and appears within the diagnosed fault list.
- It is not always possible to positively match multiple faults (C1:X3, C1:X6 has no equivalent). However, no false diagnosis was given.



*Figure 7-9 The relation between fault types which are indistinguishable when halting of primary outputs is used for diagnosis.*

Figure 7-9 shows the relation between indistinguishable fault types and their relative logical positions within the system architecture. In Figure 7-9 it is apparent that faults occurring in the function block of a specific cell are indistinguishable from faults occurring in the connections to/from the same cell. This clearly demonstrates that the maximum resolution that can be expected using this technique is to the IFIS cell level.

## 7.7 CONCLUSIONS

The purpose of this investigation was to discover the influence of the halting inherent in IFIS designs on the ability to perform diagnosis on them. The approach taken was to identify a strategy for performing fault diagnosis in IFIS designs, assess the inherent diagnostic limitations of the strategy and then discover if the strategy could be simply implemented.

Fault diagnosis was successfully performed on the FPGA implementation of the IFIS UART using only the relative halting times of primary system outputs. This demonstrates the feasibility of the technique. The advantages of using this fault diagnosis method are summarised below:

- Diagnosis can be performed using primary design inputs and outputs without internal probing.
- IFIS properties are exploited allowing the simple concept of dictionary based

diagnosis to be applied to sequential designs.

- The dictionary generation can be automated.
- The resources required to build and store the fault diagnosis dictionary are dependent on the number of IFIS cells in the system and not necessarily the complexity of the design.
- Once a dictionary has been generated for a specific design, it can be reused.
- Diagnosis can be performed by analysing the output halting sequence observed during production testing.
- Diagnosis in-the-field may be simplified because fault stimulation need not be repeated if the output halting information is available following on-line error detection. Simplifying field diagnosis can hasten repair thus increasing system availability.

The following disadvantages are associated with the fault diagnosis method:

- The highest diagnostic resolution is restricted to the IFIS cell level. For manufacturing diagnosis, this resolution is not sufficient. However, this technique may be applicable as part of a hierarchical diagnostic methodology.
- To ensure maximum diagnostic resolution for a specific design, constraints are imposed on the architecture.

While the halting based diagnosis technique may prove adequate for identifying a faulty module, a supplementary diagnosis technique that can bypass the IFIS protocol may be required if diagnosis is to be of significant use to an IC manufacturer.

# **CHAPTER EIGHT**

## **CONCLUSIONS**

### **8.1 OBJECTIVES OF CHAPTER**

This chapter concludes the thesis. It reviews the objectives of the thesis and summarises the conclusions resulting from Chapter 2 in addition to those from the investigations described in this thesis. Additionally, the success of the investigations is considered and limitations presented. This chapter concludes by suggesting ways in which this research may be extended.

### **8.2 REVIEW OF OBJECTIVES**

This thesis has assessed different aspects of the IFIS methodology. More specifically, the aims of this thesis were:

- To identify protocol/coding schemes that provide the halting mechanism characteristic of IFIS systems and to assess their impact on system level test coverage, complexity and performance.
- To discover the utility of the halting sequence information available at primary outputs for locating the source of error.

To enable the aims to be achieved, the following objectives were identified:

- To quantify the restrictions on performance and test coverage imposed by different dual-rail codes and protocols.

- To assess the characteristics of test coverage, complexity and performance associated with different circuit design techniques used to implement the chosen IFIS protocol.
- To characterise a demonstration IFIS system that incorporates multiple computation elements that are interconnected such that the IFIS protocol is enforced.
- To quantify the diagnostic resolution attainable based only on the halting sequence of primary system outputs following error detection.

In Chapter 2 design techniques developed to support on-line testing were presented in order to assess their suitability for adoption within IFIS systems. The following conclusions were drawn:

- Of coding schemes examined, none intended for on-line testing was inherently capable of detecting timing errors caused by manufacturing defects.
- Dual-rail-coding schemes developed for asynchronous designs support modular checker design and can detect such timing and unidirectional errors. The application of such schemes to on-line testing has not been characterised.
- The suitability of specific on-line test techniques that were intended for computation, to the generation of dual-rail codes was not quantified.

A series of investigations were identified in Chapter 3 to address the above issues. The outcome of these investigations was an assessment of the impact on system characteristics imposed by coding and protocol combinations supporting the IFIS halting mechanism. Additionally, conclusions could be made regarding the effects of the halting mechanism on the ability to perform failure diagnosis within IFIS systems.

### 8.3 MAIN CONCLUSIONS

This section presents the main conclusions resulting from the investigations that were performed.

System level performance can be restricted by the protocol and coding scheme: of those examined in Chapter 4, the highest performance is exhibited by inelastic protocols based on a saturated coding scheme.

The permanence of system halting is affected by the protocol and the specific nodes monitored. Chapter 4 contains evidence that to provide permanent halting requires an inelastic protocol that monitors the outputs of consecutive cells, namely: the predecessor cell, current cell, and successor cell.

The potential for the protocol and coding to restrict on-line test coverage is caused by restricting the ratio of codewords to non-codewords, thus restricting the range of vectors that can be used for on-line fault stimulation. It was shown in Chapter 4 that the ratio of codewords to non-codewords is increasingly restricted by raising the number of monitored nodes (protocol conditions) and/or by reducing the range of symbols that represent codewords.

Despite the restriction in stimuli applicable to IFIS cells during normal operation, it is possible to construct IFIS cells that exhibit high levels of on-line test coverage.

Chapter 5 provides evidence that in order to obtain a high level of on-line test coverage, while also being applicable to any digital function, the duality architecture is the most suitable for code computation in IFIS cells. Alternatives to duality are available for IFIS code computation but to gain more attractive complexity statistics than the duality architecture requires forfeiting test coverage, data throughput or generality. Within IFIS cells, a simple decoder is used for checking incoming codes and a multiplexer is used to control data-flow. This architecture is low in complexity and provides high test coverage due to the indirect observability of the decoder output by virtue of the IFIS halting characteristic.

The interconnection of IFIS cells facilitates the testing of the checker circuitry used within each IFIS cell for transmission control. This is because an artificial fault



condition (non-codeword) can be injected at the IFIS system interface, thus enabling the IFIS halting mechanism to be exercised.

The investigation described in Chapter 6 shows that by interconnecting IFIS cells such that they support the selected protocol, on-line testable systems exhibiting the register transfer level data-flow of their conventional counterparts can be realised. Furthermore, the characteristics of such systems are consistent with expectations. The ability to convert any synchronous digital system to an on-line testable equivalent using IFIS techniques makes the IFIS methodology attractive. The ability to predict the impact of such a conversion for any such system enables its feasibility to be assessed.

In Chapter 7 it was shown that the maximum resolution to which failure diagnosis can be performed using only the halting sequence of IFIS system outputs is limited to that of a single IFIS cell. This level of resolution may be further restricted if insufficient observation points are provided.

As increasingly complex systems emerge, the requirement to perform on-line testing may dominate over complexity issues. The inherent error management coupled with high levels of error coverage, low performance impact and simple failure diagnosis makes IFIS an attractive option.

#### **8.4 MEASURES OF SUCCESS**

The literature review presented in Chapter 2 showed that design techniques supporting on-line testing for protecting computation and for protecting communication have been individually developed. Coding techniques normally associated with asynchronous designs were identified as possessing characteristics that could be suitable for protecting communications between computation elements while also being suitable to system design.

The investigations were chosen to provide an understanding of the implications of adopting such a technique to provide on-line testing. The investigations have resulted in the following:

- A quantitative assessment of the restrictions on performance, behaviour and test coverage imposed by different dual-rail codes and protocols that cause halting after error detection.
- An assessment of the effects of using different techniques to design IFIS cells that support the chosen IFIS protocol. The characteristics of test coverage, complexity and performance were quantified for each technique.
- The characterisation of a demonstration IFIS system that incorporates multiple computation elements that are interconnected such that the IFIS protocol is enforced.
- A quantitative assessment of the diagnostic resolution attainable by examining only the halting sequence of IFIS system outputs following error detection.

This research can be used by system designers to assess the feasibility of implementing an IFIS version of any specific synchronous digital system. This work is of particular value because both qualitative and a quantitative approaches have been adopted to describe the advantages of the presented techniques. This thesis has provided detailed information while also placing it in the context of system design. Furthermore, it has demonstrated the developed design techniques by applying them to a hardware implementation of a complex and representative system.

## 8.5 LIMITATIONS

The following limitations apply to this research:

The UART design while complex for a small design team with strict time limitations is relatively small in comparison with the large designs appearing in the IC industry.

This effects not only the degree to which the UART can be considered representative, but also the complexity of the experimental vehicle used for failure diagnosis.

It was decided to use IFIS cell design techniques that support generic function generation, as opposed to arithmetic, iterative or Boolean calculation because of the associated simplification in design-flow. If less generic approaches had been implemented and their characteristics empirically quantified, the test coverage limitations imposed by them could have been more accurately assessed. This would have provided additional information allowing an assessment of the benefits and costs associated with using a non-generic approach.

Due to the limited number of available IFIS designs, the quantity of data representing IFIS systems does not allow predictions to be made regarding the typical maximum resolution that can be expected from fault diagnosis performed on IFIS designs.

No formal investigation to identify specific IFIS design rules that maximise the ability to diagnose failure was performed. A discussion was presented in Chapter 7 that describes the architectural influence on diagnostic resolution, however this was not developed into an algorithm that could be applied to analyse IFIS designs.

While performance characteristics were obtained by simulation, the hardware could not be stimulated at this speed. Furthermore, the characteristics of FPGAs are not necessarily representative of ASICs.

The diagnosis algorithms presented in this thesis do not consider the effects of multiple fault sources.

The complexity measurements do not take account of the additional routing imposed by the IFIS methodology.

## **8.6 FURTHER WORK**

This section presents further work that has been identified as a potentially useful extension to the current work. Furthermore, additional investigations are proposed that exhibit a strong relationship to the work described within this thesis.

### **8.6.1 Extension of Current Work**

Logic duplication was selected to generate independent data and reference information within IFIS cells. Investigations should be undertaken to explore the possibility of finding and characterising new methods for generating reference information that exploit the CMOS technology.

The maximum diagnostic resolution achieved using halting sequence is to the IFIS cell level. This can only be achieved when the IFIS architecture is constrained. Currently, IFIS designs are only confirmed as being diagnosable to this level of resolution following dictionary generation. An investigation of ways of checking and changing IFIS systems to ensure that they are diagnosable to this resolution should be undertaken.

While diagnostic resolution to the cell level is advantageous for identifying faulty ICs within an IFIS system or identifying a specific IFIS cell within an IC, higher levels of resolution are required by IC manufacturers. Therefore, further effort should be directed at investigating ways of improving the efficiency of diagnosis to the current level of resolution and exploring ways of increasing the diagnostic resolution. Resolution might be efficiently increased by combining the information available in the halting sequence with a conventional approach that circumvents the IFIS protocol.

### **8.6.2 New Investigations**

A disadvantage associated with IFIS is the complexity of resultant systems. The complexity associated with computational elements has been quantified and is

comparable with DWC. However, the area complexity associated with routing has not been quantified. Clearly, an increase in area associated with routing may increase the chances of failure. An assessment of the probability of failure associated with routing in IFIS systems would facilitate the choice of a protection scheme offering the minimum level of required protection. The implications of choosing an alternative scheme to dual-rail coding on checker complexity and ease of implementation require investigation.

In IFIS systems, handshaking provides a mechanism to control data flow following computation completion. In this thesis, coding was used to provide handshaking. In CMOS designs, current sensing offers an alternative to the examination of code sequence for providing completion detection. The application of this kind of completion detection to IFIS would remove the requirement to use communication code sequences based on monotonic transitions and may therefore permit less expensive codes to be used. The implications of using a different coding scheme would therefore require investigation.

## 8.7 SUMMARY

This chapter has presented the main conclusions that were reached as a result of the investigations described in this thesis. This serves the function of relating the intermediate conclusions to form an overall impression of what has been achieved.

In addition to combining the results from the investigations, this chapter has also assessed the degree to which the investigations have addressed the aims and objectives set out in Chapter 1 and Chapter 3. The limitations imposed by time and facilities were set out thus enabling the scope of the conclusions to be determined.

Finally, this chapter has presented a summary of the contributions made by this research concluding with suggestions for work in important related areas that would benefit from future research investment.

## REFERENCES

- [Abromov80] M.Abromovici, M.Breuer, 'Multiple Fault Diagnosis In Combinational Circuits Based On An Effect-Cause Analysis', IEEE Transactions On Computers, Vol. C-29 (6), June 1980, pp. 451-460.
- [Aitken91] R.C.Aitken, 'Fault location with current monitoring', Proc. ITC-91 Nashville October 1991 , pp. 623 - 632.
- [Aitken92] R.C.Aitken, 'Diagnosis of leakage faults with Iddq', Journal of electronic testing: Theory and applications (1992) vol. 3, pp. 367 - 375.
- [Aitken95] R.Aitken, P.Maxwell, 'Better Models or Better Algorithms ? Techniques to Improve Fault Diagnosis', Hewlett Packard Journal, Feb 1995, pp. 110 - 115.
- [Almaini89] A.E.A.Almaini, 'Electronic Logic Systems', Second Edition, Prentice-Hall, 1989, pp. 253-312.
- [Al-Saad96] H.Al-Asaad, J.Hayes, 'Design Of Scaleable Hardware Test Generators for On-line BIST', Proc. 2nd IEEE International On-Line Testing Workshop, Biarritz, 8-10 July 1996, 164-167.
- [Al-Bassam94] S.Al-Bassam, B.Bose, 'Design of Efficient Balanced Codes', IEEE Transactions on Computers, vol.43(5), March 1994, pp. 362-365.
- [Anderson73] D.Anderson, G.Metze, 'Design Of Totally Self-Checking Check Circuits for M-out-of-N Codes', IEEE Transactions on Computers. vol. 22(3), March 1973, pp.263-269.
- [Ashjaee77] M.J.Ashjaee, S.M.Reddy, 'On Totally Self-Checking Checkers For Separable Codes', IEEE Transactions On Computers, vol.. C-26(8), August 1977, pp.737-744.
- [Avizienis71a] A.Avizienis, 'The STAR (Self Testing And Repairing) Computer : An Investigation Of The Theory And Practice Of Fault Tolerant Computer Design', IEEE Transactions on Computers vol. C-20(11) Nov. 1971, pp. 1312-1321.
- [Avizienis71b] A.Avizienis, 'Arithmetic Error Codes : Cost and Effectiveness Studies For Application In System Design', IEEE Transactions on Computers vol. C-20(11) Nov. 1971, pp. 1322-1331.
- [Audet96] D.Audet, N. Gagnon, Y.Savaria 'Quantative Comparisons of TMR Implementations in a Multiprocessor System', Proc. 2nd IEEE

International On-Line Testing Workshop, Biarritz, 8-10 July 1996, 196-199.

- [Bahram92] N. Bahram, D. Chakravarty, 'ASIC Design Methodology Trends', *Electro/92*, vol 1, May 1992, pp.165-169.
- [Bartlett92] J.Bartlett et al., 'Fault tolerance in Tandem Computer Systems', pp. 586 - 648 in 'Reliable Computer Systems, 2nd Edn.', Digital Press, ISBN1-55558-075-0, 1992.
- [Berger61] J.Berger, 'A Note On Error Detecting Codes For Asymmetric Channels', *Information and Control*, Vol. 4, 1961, pp. 68-73.
- [Bernard94] J.P.Bernard, D.Durocher, 'An Expert System For Fault Diagnosis Integrated in Existing SCADA Systems', *IEEE Transactions on Power Systems*, Vol.9(1), Feb. 1994, pp. 548-554.
- [Blaum88] M.Blaum, 'Systematic Unidirectional Burst Detecting Codes', *IEEE Transactions on Computers*, vol.37(4), April 1988, pp. 453-457.
- [Bolchini95] C.Bolchini, R.Montadon, F.Salice, D.Scuito, 'Self-checking FSMs Based on a Constant Distance State Encoding', *IEEE International Workshop on Defect and Fault Tolerance in VLSI Systems*. Nov. 13-15, 1995, pp.269-277.
- [Borden82] J.M.Borden, 'Optimal Asymmetric Error Detecting Codes', *Information and Control*, Vol.53, 1982, pp. 66-73.
- [Bose85] B.Bose, D.Lin, 'Systematic Unidirectional Error Detecting Codes', *IEEE Transactions on Computers*, vol. C-34(11), Nov. 1985. pp. 1026-1032.
- [Bose86] B.Bose, 'Burst Unidirectional Error Detecting Codes', *IEEE Transactions on Computers* C35(4), April 1986, pp. 350-353.
- [Brassington95] M. Brassington, 'Semi-Custom ASIC Technology Trends', *Procs. Custom Integrated Circuits Conference 1995*.
- [Brown60] D.Brown, 'Error Detecting and Correcting Binary Codes for Arithmetic Operations', *IRE Transactions On Electronic Computers*, March 1960, pp. 333-337.
- [Brugnoni93] S.Brugnoni, G.Bruno *et al*, 'An Expert System For The Real Time Fault Diagnosis Of The Italian Telecommunications Network', *Integrated Network Management III C-12*, pp. 617-628.

- [Burns92] S.W Burns, N.K.Jha, 'A Totally Self-Checking Checker For A Parallel Unordered Coding Scheme'. VLSI Test Symposium 1992, pp. 165-170.
- [Chakra94] S.Chakravarty and S.Suresh, 'Iddq measurement based diagnosis of bridging faults in full scan circuits', 7th International Conference on VLSI design - January 1994, pp. 179 - 182.
- [Champac91] V.Champac, R.Rodriguez-Montanes, J.A.Segura, J.Figuera and J.A.Rubio, 'Fault Modelling of gate oxide short, floating gate and bridging failures in CMOS Circuits', European Test Conference, April 91, pp. 143 - 156.
- [Chang65] H.Y.Chang, 'An Algorithm for Selecting An Optimum Set Of Diagnostic Tests', IEEE Transactions On Electronic Computers, Oct. 1965. Vol. 14(5), pp706 - 711.
- [Chang70] H.Y.Chang, E.Manning and G.Metze, 'Fault Diagnosis of digital systems', Wiley-International, 1970.
- [Chang96] W.F.Chang, C.W.Wu, 'A TSC Berger-Code Checker for 2r-1 Bit Information', Proc. 2nd IEEE International On-Line Testing Workshop, Biarritz, 8-10 July 1996, 158-161.
- [Cox88] H.Cox and J.Rajski, 'A method of fault analysis for test generation and fault diagnosis', IEEE transactions on computer aided design vol. 7(7) July 1988, pp. 813 - 833.
- [David92] I.David, R.Ginosaur, M.Yoeli, 'Implementing Sequential Machines as Self-Timed Circuits', *IEEE Transactions On Computers*, Vol. 41, No. 1, January 1992, pp.12-17.
- [Dean91] M.Dean, T.Williams, D.Dill, 'Efficient Self-timing with Level-Encoded 2-phase Dual-Rail (LEDR)', *MIT Conference on Advanced Research in VLSI*, March 1991: pp. 55-70.
- [Dislis95] C.Dislis, J.H.Dick, I.Dear, A.P.Ambler, 'Test Economics and Design For Testability for Electronic Circuits and Systems', Ellis-Horwood, ISBN 0-13 108994-3,1995, pp.1-16 .
- [Dong84] H.Dong, 'Modified Berger Codes For Detection Of Unidirectional Errors', IEEE Transactions On Computers, Vol. C-33(6), June 1984, pp. 572-575.
- [Favila91] B.Rogel-Favila, A.Wakeling, P.Y.K.Cheung, 'Model Based Fault Diagnosis of Sequential Circuits and it's Acceleration', EDAC '91 (IEEE) Amsterdam 22 - 28 Feb. 1991, pp. 224 - 229



- [Ferguson91] J.Ferguson, T.Larrabee, 'Test pattern generation for realistic Bridge faults in CMOS IC's', Procs. International Test Conference 1991, pp. 492-499.
- [Frieman62] C.V.Frieman, 'Optimal Error Detecting Codes For Completely Asymmetric Binary Channels', Information and Control, Vol. 5, 1962, pp. 64-71.
- [Fujiwara84] E.Fujiwara, N.Mutoh, K.Matsuoka, 'A Self Testing Group Parity Prediction Checker And Its Use For Built In Testing', IEEE Transactions on Computers C33(6) 1984, pp. 578-583.
- [Fujiwara90] E.Fujiwara, D.K.Pradhan, 'Error-Control Coding in Computers', IEEE Computer, July 1990, pp. 63-72.
- [Garner59] H.Garner, 'The Residue Number System', IRE Transactions on Electronic Computers, June 1959, pp.140-147.
- [Genesereth84] M.Genesereth, 'The Use Of Design Descriptions In Automated Diagnosis', Artificial Intelligence, vol.24, 1984, pp. 411-436.
- [Girard95] P.Girard, C.Landrault, S.Pravossoudovitch, B.Rodriguez, 'Delay Fault Diagnosis In Sequential Circuits Based On Path Tracing', Integration, The VLSI Journal, vol.19(3), July 1995, pp. 199-218.
- [Goessel93] M.Goessel, S.Graf, 'Error Detection Circuits', ISBN 0-07-707438-6, McGraw-Hill, 1993, pp. 16-20.
- [Grant89] P.M.Grant, C.F.N.Cowan, B.Mulgrew, J.H.Dripps, 'Analogue and Digital Signal Processing and Coding', Chartwell-Bratt, ISBN 0-86238-206-8, 1989, pp. 217-232.
- [Gupta96] S.K.Gupta, D.K.Pradhan, 'Utilization of On-line (Concurrent) Checkers during Built-in Self-Test and Vice Versa', IEEE Transactions on Computers C45(1) , Jan 1996, pp.63-72.
- [Hamming50] R.W.Hamming, 'Error Detecting and Error Correcting Codes', The Bell System Technical Journal, vol. 26(2), April 1950, pp.147-160.
- [Hana86] H.H.Hana, B.W.Johnson, 'Concurrent Error Detection In VLSI Circuits Using Time Redundancy', Procs. Of the IEEE SouthEastern '86 Regional Conference 1986, pp. 208-212.
- [Hulgaard94] H.Hulgaard, S.M.Burns, G.Borriello, 'Testing Asynchronous circuits: A survey', University of Washington internal technical report 94-03-06 March 6th 1994.

- [Jha87] N.K.Jha and B Vora, 'A Systematic Code for Detecting T-Unidirectional Errors', 17th International FTC Symposium, Pittsburg, PA, July 6-8, 1987, pp 96-101
- [Johnson88] B.W.Johnson, J.H.Aylor, H.H.Hanah, 'Efficient Use Of Time And Hardware Redundancy For Concurrent Error Detection In A 32 Bit VLSI Adder', Journal Of Solid State Circuits, vol. 23(1), Feb. 1988, pp. 208-214.
- [Jones91] S.R.Jones and D.W.Lloyd, 'Digital Circuits', UK Patent application. 9225327.8, 5 December 1991.
- [Kautz68] W.Kautz, 'Fault Testing and Diagnosis in Combinational Digital Circuits', IEEE Transactions on Computers, Vol.17, April 68, pp. 352-365.
- [Khakbaz84] J.Khakbaz, E McCluskey, 'Self-Testing Embedded Parity Checkers', IEEE Transactions on Computers, C33(8) August 1984, pp. 753 - 756.
- [Ko78] D.G.Ko, M.Breuer, 'Self Checking Of Multi-Output Combinational Circuits Using Extended Parity Technique', Journal Of Design Automata & Fault Tolerant Computing, Vol.2, Jan 1978, pp. 29-62.
- [Kundu96] S.Kundu, E.S.Sogomonyan, M.Goessel, S.Tarnick, 'Self Checking Comparator with One Periodic Output', IEEE Transactions on Computers, Vol. 45, March 96, pp. 379-380.
- [Laha83] S.Laha, J.H.Patel, 'Error Correction in Arithmetic Operations Using Time Redundancy', Proc. 13th Fault Tolerant Computing Symposium, 1983, pp. 298 - 305.
- [Linder96] D.Linder, 'Phased Logic: Supporting the Synchronous Design Paradigm with Delay-Insensitive Circuitry', IEEE Transactions on Computers, vol. 45(9), September1996, pp. 1031-1043.
- [Li92] J.Li, E.Swarzlander, 'Concurrent Error Detection In ALUs By Recomputing With Rotated Operands', IEEE International Workshop Defect and Fault Tolerance In VLSI Systems, 1992, pp. 109 - 116.
- [Lo93] J.C.Lo, 'A Novel Area-Time Efficient Static CMOS Totally Self-Checking Comparator', IEEE Journal of Solid State Circuits, Vol. 28(2), Feb 1993, pp. 165 - 168.
- [Maly91] W.Maly, S.Naik, 'Defect and Design Error Diagnosability Measure', Proc. ETC-91 (Apr. 1991), pp. 83 - 90

- [Mao90] W.Mao R.K.Gulati, D.K.Goel and M.D. Ciletti, 'QUIETEST: a quiescent current testing methodology for detecting leakage faults', Proc. ICCAD-90 (November 1990), pp. 280 - 283.
- [Mead80] C.Mead, L.Conway, 'Introduction to VLSI Systems', Addison-Wesley, Reading, Ma ,1980, pp. 218 - 262.
- [Meindl87] J.D.Meindl, 'Opportunities for Gigascale Integration', Solid state Technology, December 1987, pp 85 - 89.
- [Metra96] C.Metra, J.Lo, 'Compact and High Speed Berger Code Checker', Proc. 2nd IEEE International On-Line Testing Workshop, Biarritz, 8-10 July 1996, 144-149.
- [Metra97] C.Metra, M.Favalli, B.Ricco, 'Highly Testable and Compact Single Output Comparator', Proc. 15<sup>th</sup> IEEE VLSI Test Symposium, April 1997, 210-215.
- [Millman91] S.D.Millman, E.J.McCluskey, J.M.Acken, 'Diagnosing CMOS bridging faults with stuck-at fault dictionaries', IEEE International Testing Conference 1990, pp. 860-870.
- [Millman94] S.D.Millman, J.M.Acken, 'Diagnosing CMOS bridging faults with stuck-at, Iddq, and voting model fault dictionaries', IEEE Custom Integrated Circuits Conference 1994, pp. 409-412.
- [Nigh90] P.Nigh, W.Maly, 'Test generation for Current Sensing' IEEE Design and Test, Feb. 1990, pp. 26 - 38.
- [Parekhji91] R.A.Parekhji, G.Venkatesh, S.D.Sherlekar, 'A Methodology for Designing Optimal Self-Checking Sequential Circuits', International Test Conference, 1991, pp. 283 - 291.
- [Patel82] J.H.Patel, L.Y.Fung, 'Concurrent Error Detection in ALUs by Recomputing With Shifted Operands', IEEE Transactions on Computers, C31(7) July1982, pp. 589 - 595.
- [Patel83] J.H.Patel, L.Y.Fung, 'Concurrent Error Detection in Multiply and Divide Arrays', IEEE Transactions on Computers, C32(4) April 1983, pp. 417 - 422.
- [Piestrak95] S.J.Piestrak, 'Design of self-testing checkers for unidirectional error detecting codes', ISSN 0324-9786, Oficyna Wydawnicza Politechniki Wroclawskiej, Wroclaw 1995.
- [Piestrak96a] S.Piestrak, 'Design of Self-Testing Checkers for Borden Codes', IEEE Transactions on Computers C45(4) , April 1996, pp.461-469.

- [Piestrak96b] S.Piestrak, 'Design of Minimal-Level PLA Self-Testing Checkers for m-out-of-n Codes', IEEE Transactions on Very Large Scale Integration (VLSI) Systems Vol.4(2), Jun 1996, pp.264-272.
- [Pradhan80a] D.Pradhan, J.Stiffler, 'Error Correcting Codes and Self-Checking Circuits', IEEE Computer, March 1980, pp. 27-35.
- [Pradhan80b] D.Pradhan, 'A New Class Of Error Correcting/Detecting Codes for Fault-Tolerant Computer Applications ', IEEE Transactions on Computers, Vol. C-29(6), Jun 1980. pp. 471-481.
- [Purcell88] E.Purcell, ' Fault Diagnosis Assistant', Circuits and Devices, Vol. 4(1), Jan 1988, pp 47-58.
- [Reynolds78] D.Reynolds, G.Metze, 'Fault Detection Abilities Of Alternating Logic', IEEE Transactions on Computers, vol. C-27(12). Dec. 1978, pp. 1093-1098.
- [Russell89] G.Russell, and I.Sayers, 'Advanced Simulation and Test Methodologies for VLSI Design', Van Nostrand Reinhold (International), ISBN 0-7476-0001-5, 1989, pp. 214-198.
- [Sayers85] I.L.Sayers, D.J.Kinniment, 'Low-cost residue codes and their application to self-checking VLSI systems', IEE Proceedings, vol. 132 (4), Pt. E, July 1985, pp.197 - 202.
- [Sayers86] I.L.Sayers, D.J.Kinniment, E.G.Chester 'Design of a reliable and self-testing VLSI datapath using residue coding techniques', IEE Proceedings, vol. 133 (3), Pt. E, May 1986, pp.169 - 178.
- [Siewiorek92] D.Siewiorek, R.Swarz, 'Reliable Computer Systems, Design and Evaluation', Digital Press, ISBN 1-55558-075-0, 1992.
- [Sherwani95] N.Sherwani, S.Bhingarde, A.Panyam, 'Routing in the Third Dimension from VLSI chips to MCMs', IEEE Press, ISBN 0-7803-1089-6, 1995, pp. 1-26.
- [Shedletsky78] J.Shedletsky, 'Error Correction by Alternate Data Retry', IEEE Transactions on Computers, vol. 27(2), Feb. 1978, pp.106 - 112.
- [Smith78] J.E.Smith, G.Metze, 'Strongly Fault Secure Logic Networks', IEEE Transactions on Computers', Vol. C-27(6), June 1978, pp.491-499.
- [Smith84] J.E.Smith, 'On Separable Unordered Codes', IEEE Transactions on Computers', Vol. C-33(8), Aug. 1984, pp.741-743.

- [Somayajula93] S.S.Somayajula, 'A neural network approach to hierarchical analogue fault diagnosis', IEEE Autotestcon 1993 pp. 699 - 706.
- [Sutherland89] I.E.Sutherland, 'Micropipelines', Communications ACM, June 1989, 32, (6), pp. 720 -738.
- [Thompson96] K.Thompson, 'Intel and the Myths of Test', Design and Test, Vol.13(1), Spring. 1996.
- [Varshavsky86] V.Varshavsky, 'Self-timed Control of Concurrent Processes', Kluwer Academic, 1986, pp. 309-328.
- [Waicuk89] J.A.Waicukauski and E.Lindbloom, 'Failure diagnostics of structured VLSI', IEEE Design and Test (August 1989) vol. 6, pp. 49 - 60.
- [Wakerly75] J.F.Wakerly, 'Detection of Unidirectional Multiple Errors Using Low-Cost Arithmetic Codes', IEEE Transactions On Computers, Vol. C-24(2), Feb. 1975, pp. 210-212.
- [Weste85] N.Weste, K.Eshraghian, 'Principles of CMOS VLSI Design', Addison-wesley, 1985, pp. 159-230.

## PUBLICATIONS

M.Saeed, D.Thulborn, J.Yeandel and S.Jones, 'IFIS - An On-Line Testing Methodology Using Dual-Rail Data Coding', *Proc. 2nd IEEE International On-Line Testing Workshop*, Biarritz, 8-10 July 1996, pp.68-71.

J.Yeandel, D.Thulborn, M.Saeed, S.Jones, 'Fault Localisation For On-Line Testable Designs Realised Using Dual-Rail Design Methodology', *Proc. 2nd IEEE International On-Line Testing Workshop*, Biarritz, 8-10 July 1996, pp.221-222.

J.Yeandel, D.Thulborn, S.Jones, 'An On-line Testable UART Implemented Using IFIS', *Proc. 15th IEEE VLSI Test Symposium*, California, IEEE Computer Society Press ISBN 0-81867810-0, April 27 - May 1 1997, pp. 344-349.

J.Yeandel, D.Thulborn, S.Jones, 'IFIS: An On-line Test Methodology', *IEE Circuits, Devices and Systems*, in press.

J.Yeandel, D.Thulborn, S.Jones, 'The Design and Implementation of an On-line Testable UART', *Journal of Electronic Testing: Theory and Applications (JETTA)*, accepted for publication, subject to minor revision.

