
This item was submitted to [Loughborough's Research Repository](#) by the author.
Items in Figshare are protected by copyright, with all rights reserved, unless otherwise indicated.

An assessment of dual-rail encoded on-line test methodologies and their impact on ASIC/FGPA design

PLEASE CITE THE PUBLISHED VERSION

PUBLISHER

© David Thulborn

LICENCE

CC BY-NC-ND 4.0

REPOSITORY RECORD

Thulborn, David. 2019. "An Assessment of Dual-rail Encoded On-line Test Methodologies and Their Impact on ASIC/FGPA Design". figshare. <https://hdl.handle.net/2134/13754>.

This item was submitted to Loughborough University as a PhD thesis by the author and is made available in the Institutional Repository (<https://dspace.lboro.ac.uk/>) under the following Creative Commons Licence conditions.



For the full text of this licence, please go to:
<http://creativecommons.org/licenses/by-nc-nd/2.5/>

BLDSC no:- DX 203156



Pilkington Library

Author/Filing Title THULBORN, D

Accession/Copy No.

040147270

Vol. No.

Class Mark

LOAN COPY

0401472701



BADMINTON PRESS
UNIT 1 BROOK ST
SYSTON
LEICESTER, LE7 10
ENGLAND
TEL: 0116 260 291
FAX: 0116 269 663

**AN ASSESSMENT OF DUAL-RAIL ENCODED
ON-LINE TEST METHODOLOGIES AND THEIR
IMPACT ON ASIC/FPGA DESIGN**

by

David Thulborn, B.Sc., M.Sc.

A Doctoral Thesis.

Submitted in partial fulfilment of the requirements

for the award of


Doctor of Philosophy

of

Loughborough University.

March 1997

© by David Thulborn 1997

 Loughborough University	
Faculty of <input type="text"/>	
Department of <input type="text"/>	
Date	Oct 97
Class <input type="text"/>	
Acc No.	080 147 270

99100370

ABSTRACT

The testing of fabricated Integrated Circuits (IC's) is of great concern to production engineers and circuit designers alike. With the complexity of Very Large Scale Integrated (VLSI) circuits increasing every year, the problem of testing the fabricated designs is becoming acute. Several methods for reducing the burden of IC testing have been incorporated into the designs being tested thus giving rise to the phrase Design For Test (DFT).

This thesis aims to understand how dual rail encoding of digital data can affect the different characteristics of electronic circuits. More specifically, it investigates a novel on-line test methodology called IFIS (If it Fails, It Stops), and its impact upon the design and implementation of electronic circuits intended for Application Specific Integrated Circuit (ASIC) and Field Programmable Gate Array (FPGA) technologies.

The first two studies investigate the characteristics of the IFIS methodology to determine the most efficient and effective encoding scheme, protocol rules and feedback structures required for data processing. The third study investigates a series of possible improvements to the design of IFIS cells and determines the most efficient method of designing cells using the IFIS methodology. The final study investigates the feasibility of IFIS using a 'real life' commercial UART re-engineered using the IFIS methodology.

The outcome of this work is an identification and characterisation of the factors which influence the performance and implementation cost of the IFIS methodology.

ACKNOWLEDGEMENTS

I wish to take this opportunity to offer my sincere thanks to my supervisor Professor Simon Jones for his support, criticism, encouragement and technical guidance throughout this period of research. I shall always be grateful to him.

I also wish to thank all the members both former and present, of the Electronic Systems Design Group based at Loughborough University, for their help and encouragement. They have been the source of many useful and stimulating discussions. In particular, I wish to thank Mr. Mark Gooch and Mr. Julian Yeandel in this respect. I would especially like to thank Mr. Julian Yeandel for producing the top level structure of the 'C' model used for the verification of the UARTs.

To my parents I extend my warmest thanks. I wish to thank them for the education they have afforded me and for their support and encouragement over the last 3 years. Also, I must send a very special thank you to my girlfriend Claire and son Michael for always being there to see me through.

For my financial support I wish to thank the Engineering and Physical Sciences Research Council, without whom this work would not have been possible.

Finally, I wish to send a special thank you to Mr. Steve Stavrou whose enthusiasm and humour have kept me motivated.

TABLE OF CONTENTS

	Page
Abstract	i
Acknowledgements	ii
Statement of Originality	iii
Table of Contents	iv

CHAPTER ONE

INTRODUCTION	1
1.1 Introduction	1
1.2 Motivation	2
1.2.1 Off-line Testing	3
1.2.2 On-line Testing	5
1.3 Aims of the Thesis	7
1.4 Structure of the Thesis	8

CHAPTER TWO

REVIEW	10
2.1 Objectives of Review	10
2.2 Motivation	10
2.3 Hardware Redundancy	12
2.3.1 Duplication With Comparison	12
2.3.2 Triple Modular Redundancy	13
2.3.3 N Modular Redundancy	13
2.3.4 Summary of Hardware Redundant Techniques ...	14
2.4 Time Redundancy	15
2.4.1 Alternating Logic	16

2.4.2	Recomputing With Shifted Operands	16
2.4.3	Recomputing With Swapped Operands	17
2.4.4	Summary of Time Redundant Techniques	18
2.5	Information Redundancy	19
2.5.1	Duplication Codes	21
2.5.1.1	Alternate Data Retry	21
2.5.1.2	Summary of Duplication Codes	22
2.5.2	Arithmetic Codes	22
2.5.2.1	Non-Separable Arithmetic Codes	22
2.5.2.2	Separable Arithmetic Codes	23
2.5.2.3	Summary of Arithmetic Codes	25
2.5.3	Unidirectional Error Detecting Codes	25
2.5.3.1	Unordered Codes	26
2.5.3.1.1	Non-separable Unordered Codes	27
2.5.3.1.2	Separable Unordered Codes	27
2.5.3.2	t-Unidirectional Error Detecting Codes	28
2.5.3.2.1	Non-separable t-Unidirectional Error Detecting Codes	28
2.5.3.2.2	Separable t-Unidirectional Error Detecting Codes	29
2.5.3.3	Burst Unidirectional Error Detecting Codes	30
2.5.3.3.1	Separable Burst Unidirectional Error Detecting Codes	31
2.5.3.4	Summary of Unidirectional Error Detecting Codes	33
2.5.4	Parity Based Codes	34
2.5.4.1	Bit-per-Word Parity	34

2.5.4.2 Interlaced Parity	35
2.5.4.3 Parity Check Matrix	35
2.5.4.4 Summary of Parity Based Codes	36
2.5.5 Summary of Information Redundant Techniques .	36
2.5.6 Totally Self Checking Checkers	39
2.5.7 Partially Self Checking Checkers	40
2.6 Summary	40
2.7 Conclusions	42

CHAPTER THREE

OVERVIEW OF INVESTIGATIONS	43
3.1 Objectives of Chapter	43
3.2 Aims of Research	43
3.2.1 Statement of Research Objectives	44
3.3 Experimental Vehicles	45
3.3.1 FIR Filter	45
3.3.2 UART	45
3.3.3 System Environment	46
3.4 Introduction to Investigations	47

CHAPTER FOUR

IFIS ON-LINE TEST METHODOLOGY	49
4.1 Objectives of Chapter	49
4.2 Motivation	49
4.3 IFIS Encoding Schemes	50
4.3.1 IFIS 1 Encoding Scheme	50
4.3.2 IFIS 2 Encoding Scheme	51
4.4 IFIS Processing Methods	52

4.4.1 Elastic Processing	53
4.4.2 In-Elastic Processing	54
4.4.3 Data Driven Processing	56
4.5 IFIS Feedback Structures	57
4.5.1 Self Feedback	58
4.5.1.1 In-elastic Processing	59
4.5.2 Successor Feedback	60
4.5.2.1 Elastic Processing	61
4.5.2.2 In-elastic Processing	62
4.5.2.3 Data Driven Processing	63
4.5.3 Successor Successor Feedback	63
4.5.3.1 Elastic Processing	64
4.5.3.2 In-elastic Processing	65
4.5.3.3 Data Driven Processing	65
4.5.4 Successor and Self Feedback	67
4.5.4.1 Elastic Processing	68
4.5.4.2 In-Elastic Processing	68
4.5.4.3 Data Driven Processing	69
4.6 Summary of Possible IFIS systems	70
4.7 Conclusions	73

CHAPTER FIVE

IFIS IMPLEMENTATION STUDY.....	75
5.1 Objectives of Chapter	75
5.2 Motivation	75
5.3 IFIS Design Implementation Study	76
5.3.1 FIR Filter	76
5.4 IFIS FIR Filter Comparisons	77

5.5 Conclusions	82
 CHAPTER SIX	
IFIS CELL DESIGN STUDY	84
6.1 Objectives of Chapter	84
6.2 Motivation	84
6.3 IFIS Cell internals	85
6.3.1 IFIS Control Block	86
6.3.2 Multiplexor Block	88
6.3.3 Storage Block	88
6.3.4 Functional Block	89
6.4 Design Solutions	89
6.4.1 Behavioural Synthesis	90
6.4.2 Arithmetic Coding	91
6.4.3 Duality	93
6.4.4 Time Redundancy	94
6.5 Summary of Results	96
6.6 Conclusions	97
 CHAPTER SEVEN	
IFIS FEASIBILITY STUDY	99
7.1 Objectives of Chapter	99
7.2 Motivation	99
7.3 Design Partitioning of the IFIS UART	100
7.3.1 UART Controller Design	101
7.4 UART Verification	102
7.5 Fault Injection	104
7.6 The Impact of The IFIS Methodology	107

7.7 Scalability of the IFIS Methodology	109
7.8 Summary of IFIS UART Case Study	110
7.9 Conclusions	111
 CHAPTER EIGHT	
CONCLUSIONS	112
8.1 Objectives of Chapter	112
8.2 Review of Objectives	112
8.3 Experimental Investigations.	113
8.4 Main Conclusions	114
8.5 Measures of Success	116
8.6 Limitations of the Work	117
8.7 Further Work	118
8.8 Summary of Thesis	118
 References	119
Bibliography	125
Published Papers	126

CHAPTER ONE

INTRODUCTION

1.1 Introduction

Integrated circuit technology currently enables highly complex circuit designs containing hundreds of thousands of gates to be realised. Along with the increase in gate count has come a decrease in gate cost and improvements in performance. The ability to integrate several millions of transistors into a monolithic integrated circuit (IC) offers many advantages. However, a major concern in the production of these complex devices is the ability to verify their fault free operation. The testing of digital circuits has been difficult since the beginning of the electronics industry. The problems associated with test have been aggravated by several effects arising from the increase in complexity and reduction in the pin to gate ratio. These include:

- Technological advances have enabled the prolific use of complex devices in many applications ranging from consumer products to critical commercial controllers and consequently reliability is highly important.
- The widespread deployment of ICs means reliability is an important attribute for electronic systems (Williams (1983)). Williams states that the cost of detecting a fault increases by a factor of ten from chip to board, board to system and system to 'in the field' fault.

Commercial applications are demanding higher levels of integration and higher levels of reliability. These demands need to be met with high levels of testing. Fault

coverages of greater than 99% are routinely expected during the testing of complex ASIC devices as noted in Bahram (1992).

1.2 Motivation

The objectives in testing an electronic design are twofold:

- To ensure that the pre-fabrication circuit behaviour satisfies the intent of the designer and it is free from timing and functional design errors.
- To detect post-fabrication faults.

The testing of devices after fabrication is one of the most difficult problems confronting designers and test engineers as noted by Williams (1983). The problems met are:

- Test pattern generation and evaluation times are increasing with circuit complexity.
- An increase in the volume of test data needed, namely the number of input vectors and output response pairs required to test the circuit.
- The increased requirement for complex devices has driven the development of more sophisticated CAD tools. However, the greatest advances have been in layout and simulation areas and less of an improvement in the area of test until recently (Bahram (1992)).
- The number of possible faults that have to be considered is large, as complex circuits can contain substantial numbers of components (e.g. memory elements, gates and interconnect lines) which are individually subject to different kinds of faults.
- The observability and controllability of the internal elements of any circuit are limited by the number of available input/output (I/O) pins. As chip complexities increase, the task of creating an adequate test to control or observe a single circuit node becomes more difficult.

- The high speed nature of contemporary devices requires high-speed test systems that can test the circuits when they are operating at their maximum speeds.

Solving the problems above increases the number of test patterns required for a successful test. This in turn increases the time required for applying the test and the computing resources required to store the test patterns and the results. Moreover, Williams (1983) shows that it has now become impracticable to attempt fully exhaustive testing of complex circuits. As a result, several alternative test methods have been suggested to alleviate the problem. The testing solutions for electronic systems fall broadly into two categories, off-line and on-line testing.

1.2.1 Off-Line Testing

Off line testing, or explicit testing, separates the testing process from the normal operation process. In general, off line testing involves three steps:

- Generation of the test patterns
- Application of the test patterns,
- Evaluation of the responses obtained from the device under test

The goal of the test pattern generation phase is to produce those input patterns which will exercise the Device Under Test (DUT) in different modes of operation while trying to detect any existing faults. The production of the test patterns is a difficult task as each fault considered must be excited and then propagated to a primary output. A wide range of algorithms have been proposed to automate the test pattern production stage at both gate and transistor level, while targeting many different fault types (e.g. Aitken (1991), Bleeker (1993), Corno (1996), Goel (1981), Fujiwara (1983), Reddy (1985), Schulz (1989), Larrabee (1989), Nigh (1990), McEuen (1996) and Ferguson (1991)). It has been shown (e.g. Williams (1983) and Seth (1985)) that

computer run time to perform test generation is approximately proportional to the cube of the number of gates within the design. As circuit complexity increases, this overhead becomes more intractable, making test pattern generation increasingly complex.

The application of test patterns can be accomplished in two ways, either via external testing or through the use of Built In Self Test (BIST) structures within the design. With external testing, Automatic Test Equipment (ATE) is used to apply the test patterns externally. This gives good control over the test process and allows testing under different timing and electrical conditions.

With BIST techniques, the DUT is forced to execute a self test procedure. This is less expensive in terms of required test equipment, but requires greater design effort to incorporate the BIST structures into the design. There are numerous techniques for designing BIST structures put forward by Asaad (1996), Konemann (1979), Beucler (1984), McCluskey (1985), Avra (1993), Kothari (1993), Fertsch (1991) and Zorian (1993). These techniques have been categorised under the Design For Test (DFT) heading and aim to aid the testing of circuits using a structured design manner. These DFT methods have a variety of performance and complexity overheads associated with each implementation, and will not be discussed within the framework of this thesis. For further information see Golstein (1979), Haider (1993), Hellerbrand (1996), Abadir (1983) and Muehldorf (1981).

The process of evaluating the responses from the device under test usually aims to achieve two goals. Firstly, to distinguish between the good and faulty devices, and secondly, to provide fault location information for diagnostic purposes. There are several techniques available to aid in the evaluation of the response vectors from the device under test. All of these use some form of comparison between a known good, or 'Golden Signature' response, and the actual response from the device under test. This leads to a go/no-go decision as to the reliability of the device.

Generally, off-line testing has the following advantages:

- The test vectors, once generated, can be applied at speed to the DUT. This allows full testing of the DUT's electrical and timing responses.
- The fault coverage of the vector set can be determined with a high degree of accuracy and test vectors generated until the desired level of fault coverage is achieved.

However, there are also several disadvantages with off-line testing, namely

- Test pattern generation requires considerable amounts of computing resource to generate vectors and to store the stimulus vector set.
- Exhaustive testing is impractical.
- Fault simulations are required to determine the level of fault coverage a particular vector set achieves. This also requires considerable amounts of computing resource.
- Faults occurring after production tests cannot be detected (e.g. transient faults, stress faults, wear out faults).

These problems are going to become increasingly apparent as circuit complexity grows and device dimensions shrink.

1.2.2 On-Line Testing

On-line testing, or implicit testing, integrates the testing process with the normal operation of the device. Any device designed using on-line test concurrently tests itself during normal operation. This generally requires the introduction of redundancy into the circuit design. This redundancy usually takes the form of either information redundancy, time redundancy or hardware redundancy. In practice, one type of redundancy (which is predominant) usually involves introducing some other type as well. As an example, consider information redundancies such as encoding of data with

an error detecting code (EDC). This relies on introducing extra bits to any data unit, e.g. by encoding inputs and outputs of all circuits (and internal states if the circuit is sequential) with EDCs. However, it also involves hardware redundancy such as encoders, decoders, extra data path lines (to transmit) or extra memory elements (to store) these extra bits. Time redundancy is also introduced as the encoding / decoding of data may introduce some delay into the system. Although the introduction of extra redundancy is required, the benefits when compared to explicit testing can be attractive:

- Explicit testing expenses (e.g. for test equipment, down time, and test pattern generation) are eliminated during the life of the system as the data patterns used in normal operation serve as test patterns.
- The faults are detected instantaneously during the use of the chip, hence the first faulty data pattern caused by a certain fault is detected. The user can then rely on the output results within the degree of fault coverage provided by the error detection method used. In explicit approaches, nothing can be said about the correctness of the result until the chip is fully tested.
- Transient faults, which may occur during normal operation, are detected if they cause a faulty data pattern. These faults cannot be detected by any explicit testing method.

Unfortunately, the on-line testing approach also suffers from several drawbacks that limit its usage in testing:

- The data patterns (application patterns) may not exercise all the storage elements or all the internal connection lines within the chip. Defects may exist in places not exercised, and hence the faults produced will not be detected.
- Using information redundancy to code the information used in an IC often requires additional I/O pins. At least two extra pins are needed as error signal indicators (A single pin cannot be used, since a single pin stuck at the

good value could go undetected). Because of the constraint on pin count, however, such requirements cannot easily be fulfilled.

- Additional hardware circuitry is required to implement checkers, storage units and to transmit any coded information.
- Designing a circuit for on-line testing is often a more complex task than designing a similar circuit that is tested explicitly.
- On-Line test approaches provide no control over critical voltage or timing parameters. Hence, devices cannot be tested under marginal timing and electrical conditions.
- The degree of fault coverage usually provided by implicit methods is less than that provided by explicit methods.

The on-line and off-line methods of testing digital electronic circuits both have a common aim - to distinguish between good and faulty devices in the most cost effective manner. In Williams (1983), it is shown that the greatest commercial cost occurs if a device fails in the field, which is **after** passing its production test. It has also been shown by Sieviorek (1992) and Jha (1993) that transient faults are by far the most common types of faults occurring during the operation of contemporary electronic systems and it is these types of faults that off-line testing is unable to detect. Consequently, although off-line testing is considered to be the more comprehensive test since the fault coverage is generally higher, only on-line testing can offer any form of protection against the more expensive 'in-the-field' component failures a company may face. In addition to this, many designs which incorporate on-line test strategies also undergo an off-line test immediately after fabrication. Consequently, on-line test is an exciting area of research that is currently exhibiting rapid growth.

1.3 Aims of the Thesis

This thesis focuses on a novel approach to on-line testing under development at Loughborough University. The approach, IFIS (If it Fails, It Stops), is a test methodology that uses dual-rail encoding of data and handshaking between

computation elements to achieve on-line test. This thesis attempts to evaluate the IFIS on-line test methodology and offer evidence as to its suitability for IC design. The evaluation of the IFIS methodology is achieved by employing two aims, namely

- An identification and assessment of the encoding schemes, fundamental building blocks and design methodologies for IFIS.
- An evaluation of the utility of the IFIS concept issues through the construction of a 'realistic' demonstrator IC.

1.4 Structure of the Thesis

A review of related work is provided in **Chapter Two**. Firstly, the current state of on-line testing is examined within the context of information, time and hardware redundancy. The review shows how the various on-line test techniques are related and compares the overheads associated with each method. Comparisons of the various on-line test methods help to identify areas of interest; in particular it focuses on IFIS.

Chapter Three introduces the objectives of the experiments, used to fulfil the aims of the thesis. It outlines four experiments to satisfy the objectives and explains the experimental assumptions used in these investigations.

Chapter Four discusses the first experiment which provides an in-depth investigation into the IFIS methodology with the goal of identifying the required characteristics for processing data. This involves a study into encoding schemes, processing rules and of feedback structures within an IFIS system. The outcome of the experiment identifies effective encoding schemes and feedback structures for use within the IFIS methodology.

Chapter Five details the implementation impact of the IFIS methodology when targeted to ASIC and FPGA technologies. The different encoding schemes and

feedback systems undergo hardware implementation to identify and assess the more cost-effective combinations.

Chapter Six details a design study focused on producing a less complex cell design for the IFIS methodology. Several possible design solutions undergo hardware evaluation to identify the most generic and cost-effective cell design for IFIS cells.

Chapter Seven focuses on implementing the new methodology into a 'real life' design as a feasibility study. A commercial UART (Universal Asynchronous Receiver Transmitter) is re-engineered using the IFIS methodology. Comparisons are then drawn between the IFIS and conventional UARTs to highlight the impact of the methodology on 'real life' designs.

Chapter Eight draws together the conclusions from each of the four investigations and discusses whether the stated objectives have been achieved. It examines the limitations of the work and outlines possible extensions to the existing investigations, as well as further research areas that may be of interest for extended study. Finally it summarises the main points of the thesis.

CHAPTER TWO

REVIEW

2.1 Objectives of Review

The main objectives of this chapter are three fold:

- To review the current state-of-the-art in on-line testing,
- To identify interesting topics which could be investigated further,
- To specify the area in which the work in this thesis will address, given the available time and resources.

With these objectives met, the experiments performed within this thesis should be in context.

2.2 Motivation

This work is necessary to help identify areas that have not been fully explored and to help clarify the direction taken by the field in general. The field of on-line test can be broken into three categories as shown in Figure 2.1:

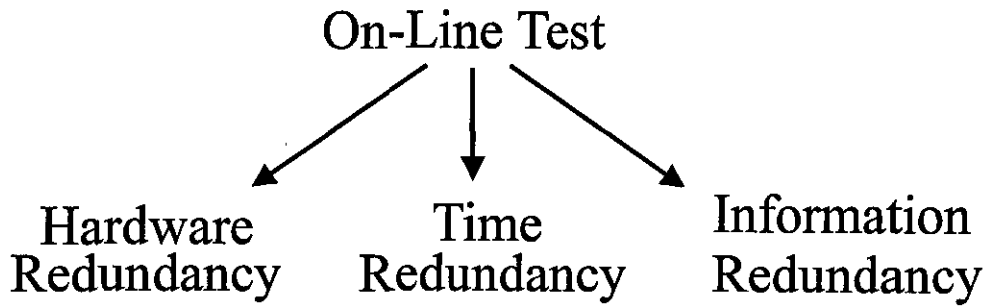


Figure 2.1 - Redundancy Methods within On-Line Test

- **Hardware Redundancy:** This technique is one of the simplest, requiring multiple copies of the design to be placed in parallel. The circuit outputs are compared after each computation and if no errors have occurred, the results will be identical. In the presence of a fault, the faulty design will yield an output inconsistent with the other copies. These methods are discussed in section 2.3.
- **Time Redundancy:** This technique involves the re-use of a single hardware design at successive points in time. The data is applied, the computation is performed, and the output response is stored for comparison with another calculation using the same data. If no errors have occurred, the results will be identical. However, if a fault has occurred then the results will be inconsistent. Several methods have been presented in the literature to prevent a fault from affecting both computations in the same manner, thus yielding a corrupted but undetected output response. These are discussed in section 2.4.
- **Information Redundancy:** The final technique uses redundancy within the transmitted data to determine if a fault is present. The information is encoded before operation, and decoded after operation. Any fault affecting the operational data alters the expected codeword and is detected. A variety of coding schemes exist which offer different degrees of protection against faults. These schemes are discussed in section 2.5.

Each redundancy technique brings benefits and penalties which make it suit a particular implementation. This review discusses the advantages and disadvantages of

each technique, and highlights areas where further work could be undertaken. This allows the direction of the work within this thesis to be identified. The three categories of hardware, time and information redundancy described above will now be reviewed further.

2.3 Hardware Redundancy

Several forms of Hardware Redundancy exist which are all based on an idea originally proposed by von Neumann in 1956. The idea was to replicate the original circuit such that a single fault would affect only one copy of the circuit in question. The remaining correctly operating circuits would thus mask the fault and still produce a correct output. Although this basic technique is simple and easy to achieve, it is also limited by the addition of potentially large amounts of redundant hardware. Unfortunately, along with the increase in complexity comes an increase in power consumption and a substantial increase in the physical size occupied by the extra hardware. These factors can make the use of hardware redundancy very difficult to justify. Several variations of von Neumann's proposal have been implemented, with varying degrees of success.

2.3.1 Duplication With Comparison

One form of hardware redundancy is that of duplication with comparison. This is the simplest form of von Neumann's idea and requires the duplication of the circuit as described in Johnson (1988). The outputs of the two circuits are fed into a comparator which determines if a fault has occurred. Duplication with comparison involves a simple and straightforward design style that is effective in detecting all single stuck-at faults which result in an error. Unfortunately, duplication with comparison requires careful design of the comparator to ensure that any faults within the comparator do not cause an incorrect output response or mask a fault within the duplicated units (i.e. the comparator must be self-checking). Another disadvantage is the increase in physical size and power consumption of the design. Although the increased circuit complexity

could possibly be tolerated, the increased power consumption and weight of this technique are considerable disadvantages.

2.3.2 Triple Modular Redundancy

Another simple form of hardware redundancy is that of Triple Modular Redundancy (TMR) as described in Russell (1989) and Audet (1996). The logic circuits are triplicated and fed into a majority voting element which usually calculates the majority function $M = XY + XZ + YZ$. The logic circuits are not restricted in terms of complexity, and can be any design from a simple gate to a microprocessor or larger. The majority function ensures that if one circuit is faulty, the other two circuits will override the error and mask the fault. However, there are two main problems with this type of on-line error detection scheme. Firstly, two circuits failing in the same manner will produce an incorrect output response. Secondly, the majority voting element is a single point of failure. Consequently, although the voting element is usually much simpler than the circuits being triplicated, its design is of the utmost importance. The voting element must be the most reliable unit, as a single fault in the voter can still produce an incorrect output response. In an attempt to overcome this problem, the N-Modular Redundancy technique was proposed.

2.3.3 N-Modular Redundancy

A method for fault tolerance has also been described in Russell (1989) where the technique of TMR was expanded to that of N-Modular Redundancy (NMR). The main difference between NMR and TMR is that the function circuit and the voting elements are both duplicated N times. This gives greater tolerance to faults as $(N+1)/2$ circuits would need to fail in exactly the same way for an incorrect output response to be generated. Statistically, the probability of the $(N+1)/2$ circuits all developing the same fault at the same time is considerably small. The single point of failure in TMR, the

voting element, is also duplicated N times. This eliminates the possibility of a single fault in the voting circuit causing an incorrect output response to be generated. A reliability analysis of systems using both TMR and NMR is given in Russell (1989).

2.3.4 Summary of Hardware Redundant Techniques

The main penalties paid when using the hardware redundant techniques are the overheads created during implementation. Duplication requires at least twice the overhead (two circuits and a comparator), TMR requires at least three times the overhead (three circuits plus the voting element), while NMR requires more than N times (N circuits plus the N voting elements). In addition to this, duplication with comparison and triple modular redundancy both have single points of failure (i.e. the comparator or voting element) which requires careful design to ensure the hardware redundancy is not rendered in-effective by a fault within the comparator. These points are highlighted in Table 2.1. The points shown in Table 2.1 are significant disadvantages as some of the implementation characteristics are highly undesirable. For many applications which use ASIC technologies, the increase in complexity and power cannot be justified and consequently a more efficient means for implementing on-line test needs to be found.

Redundancy Method	Area Overhead	Power Consumption	I/O Pins	System Size	Single Point of Failure ?
Duplication with Comparison	> x 2	> x 2	> x 2	> x 2	Yes, Comparator.
Triple Modular Redundancy	> x 3	> x 3	> x 3	> x 3	Yes, Voting Element.
N-Modular Redundancy	> x N	> x N	> x N	> x N	No.

Table 2.1 - Hardware Redundancy Summary

2.4 Time Redundancy

The hardware redundant methods for on-line testing require an increase of at least 100% (duplication with comparison), 200% (triple modular redundancy) or greater (n-modular redundancy) in terms of area, power consumption, weight, size and overall system cost. In an attempt to overcome some of the difficulties with hardware redundancy, time redundancy has been proposed and has received much attention (e.g. Reynolds (1978), Patel (1982) and Hana (1986)). Using redundancy in time also leads to the advantage that on-line checking of the circuit operation is achieved without increasing the number of I/O pins. In complex designs, where constraints on I/O pin counts can be of great importance, time redundancy could thus be an attractive approach. The main areas involving time redundancy that have received recent attention will now be explored.

2.4.1 Alternating Logic

A design technique which uses time redundancy has been proposed by Reynolds (1978) where the successive execution of a required function and its dual is used to determine the validity of the output response. This technique requires all circuits designed in this manner to exhibit the 'self-dual' property. The 'self-dual' property is difficult to achieve as it requires a circuit to have the following property:

If a combinational circuit implements the Boolean function $Y = F(X_1, X_2, X_3)$, then the function $F(X_1, X_2, X_3)$ is self-dual if inverting all of the inputs yields an inverted output, such that $\bar{Y} = F(\bar{X}_1, \bar{X}_2, \bar{X}_3)$.

Evidently any arbitrary function is not self-dual and cannot be immediately implemented using this design style. However, according to Reynolds (1978), it is possible to convert any non-self-dual function of n variables into a function of $n+1$ variables that is self-dual. The converted circuit can then be implemented using this technique. The self-dual method is also adaptable to sequential circuitry, but does require the addition of memory elements within the design. Unfortunately, transforming an arbitrary function into a self-dual equivalent usually incurs a substantial increase in the amount of circuitry required for the implementation. Johnson (1988) notes that the alternating logic technique may require an increase of 85%-100% in hardware to make a function self-dual. This technique is then comparable to duplication with comparison.

2.4.2 Recomputation With Shifted Operands

Another method of concurrent error detection called Recomputing with Shifted Operands (RSO) was proposed by Patel (1982). This technique applies an input data word to an arithmetic unit and calculates the result. The result is then shifted N bit positions and stored. The inputs are shifted M bit positions and re-applied to the same unit (e.g. for addition $M=N$, and for multiplication, $M^2 = N$). Under fault free

conditions the two results will be identical, as the shift is present only to force different data paths through the arithmetic unit. This ensures a single fault will not affect both calculations in the same manner and does not affect the arithmetic operation performed by the unit. The complexity increase caused by the implementation of the RSO technique depends upon the original circuit, but has been shown by Hana (1986) to be 31.4% for an ALU. A similar technique of recomputing with rotated operands (RRO) has also been proposed by Patel (1982) and separately by Li (1992). With this technique, the operands are applied, barrel shifted and then re-applied in an attempt to detect errors. Similarly, this approach uses barrel shifting in an attempt to force different data paths through the design, and does not affect the arithmetic operation taking place. The complexity increase for this technique is similar to the RSO overhead, as RRO is a special case of RSO as noted by Patel (1982). As both the RSO and RRO techniques effectively apply different vectors to the functional circuit during the second computation, the corresponding fault coverage is considerably higher than the hardware redundancy techniques of duplication with comparison and TMR as noted in Johnson (1988). Unfortunately, both approaches only work for certain arithmetic functions (e.g. addition and multiplication) and cannot be applied to arbitrary logic functions. Although these methods do allow the detection of single faults within the arithmetic unit, the inability to implement the technique to functions other than simple arithmetic is a major disadvantage.

2.4.3 Recomputation With Swapped Operands

Recomputation with swapped operands (RSWO) is a variation of the recomputation with shifted operands (RSO) as shown in Hana (1986). The encoding and decoding function is that of swapping the upper and lower halves of each operand. At time t_0 , the computation is performed using the unmodified operands and the result is stored for later comparison. At time $t_0 + \delta t$, the upper and lower halves of each operand are swapped and the computation repeated. This method has the advantage of being simple to implement, while only requiring a complexity increase of 23.8% for the same ALU implementation discussed for RSO (Hana (1986)). The advantages of this

approach are that the hardware overhead required to implement the required swapping is 5%-10% less than that required by the shifting operand technique. However, the disadvantages of this technique are that the comparison stage still requires careful design such that the extra redundancy is not rendered in-effective by a fault within the comparator itself. Consequently, the comparator must again be self checking.

2.4.4 Summary of Time Redundant Techniques

The technique of time redundancy immediately brings a reduction in data throughput, as 50% of the calculations are used solely for the purpose of error detection. The time redundant methods discussed also come with some form of hardware redundancy, either in the form of extra registers required to hold a result before comparison, the comparison hardware, or the extra bits since the data word width is increased (e.g. due to shifting the operands in RSO). Although this extra hardware is usually smaller than the various overheads of hardware redundancy, it can still be significant with Alternating Logic requiring up to 85% as stated by Johnson (1988). As with hardware redundancy, careful design of the comparison hardware must be undertaken so that a fault within the comparator does not render the time redundancy in-effective. These points are highlighted in Table 2.2. From Table 2.2, it can be seen that although time redundancy does have some advantages when compared to hardware redundancy, such as decreased hardware overhead and lower I/O pin counts, it also has some disadvantages which need to be considered. These disadvantages include a still significant hardware overhead and reduced data throughput. In many ASIC designs, where operational speed is crucial, the reduced data throughput incurred by time redundancy could severely limit the performance of the system.

Redundancy Method	Data Throughput	Hardware Overhead	Increase in I/O Pins ?	Single Point of Failure ?
Alternating Logic	50%	85-100%	Negligible	Comparator
Recomputing with Shifted Operands	50%	$\approx 31.4\%$	Negligible	Comparator
Recomputation with Rotated Operands	50%	$\approx 40\%$	Negligible	Comparator
Recomputing with Swapped Operands	50%	$\approx 23.8\%$	Negligible	Comparator

Table 2.2 - Time Redundancy Summary

2.5 Information Redundancy

Both of the redundancy techniques previously discussed can incur substantial penalties in terms of hardware, I/O pins and power consumption. This can make them less attractive to ASIC implementation. The third technique in the on-line testing environment is information redundancy. Information redundancy is the process whereby the data to be manipulated is coded such that any faults within the system alter the expected code and are detected. Data coding introduces redundancy into the data stream which usually manifests itself in terms of extra bits in the data word being manipulated. These extra bits provide the means for the receiver to determine if an error has occurred and take the appropriate action. There are various techniques within the on-line test environment which use data coding, and these can be broken into four coding groups as shown in Figure 2.2.

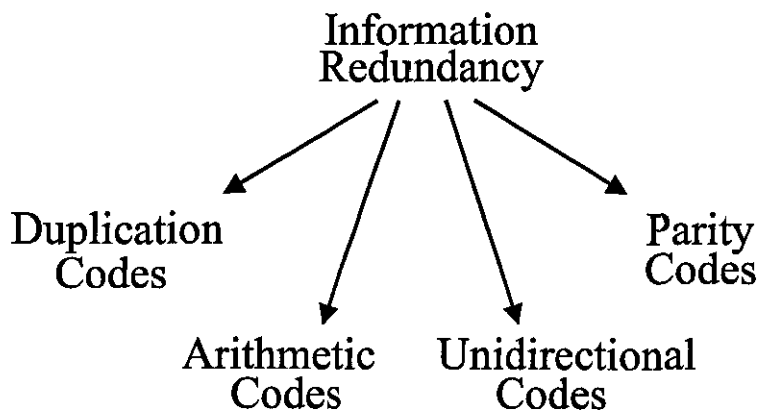


Figure 2.2 - Information Redundant Code Categories

- **Duplication Codes:** This coding technique duplicates the entire word to be transmitted, with bit by bit inversion in the duplicated word.
- **Arithmetic Codes:** These coding techniques are designed specifically for arithmetic operations. Here the coding function is retained or modified in a known manner through the arithmetic operation. Some of these codes can be used for logical operations as well.
- **Unidirectional Error Detecting Codes:** These codes are designed specifically to offer protection against faults which manifest themselves as single, multiple or burst bit errors in the '0' → '1' direction, or the '1' → '0' direction, but not both simultaneously.
- **Parity Based Codes:** These codes are based purely on the generation and checking of parity bits.

Unfortunately, selection of only one error detection code that would be best suited for a particular digital system is infeasible. It is well known that the error codes suitable to protect bus data or the memory system are not preserved by the arithmetic circuitry. Conversely, error codes suitable to protect arithmetic circuitry may be too expensive to be used for data bus protection and not powerful enough to be used in memory systems (which generally require the use of an error correcting code). Consequently, all four classes of error detecting codes are required depending upon the application and consequently will now be reviewed in more detail.

2.5.1 Duplication Codes

Duplication codes are the most redundant error detecting codes available with 100% redundancy, but also have the advantage that totally self checking circuits (see section 2.5.6) are generally simple to construct. Methods for translating duplication codes directly into TSC circuits are readily available in Ashjaee (1977), Nikolos (1996) and Piestrak (1995).

2.5.1.1 Alternate Data Retry

One information redundancy technique that uses duplication coding is that of alternate data retry as described in Shedletsky (1978). The alternate data retry method is a re-execution of an operation which initially fails to produce an error-free result. The alternate data retry method uses an alternative data representation in an attempt to nullify the effects of any faults. The choice of alternate data representation and the design of the processing circuits combine to ensure that even an error due to a permanent fault is not repeated during retry. This is best illustrated with the example from Shedletsky (1978), where unit A sends an odd-parity vector <0001> over the bus to unit B. Due to the presence of a stuck-at fault, unit B receives the erroneous, even-parity vector <1001> and signals an error. For the retry, unit A sends the alternate data representation <1110>, which is received correctly despite the fault. In this example, both the vectors <0001> and <1110> represent the same message and convey the same information between units. This system of recomputing a result if an error condition occurs involves information redundancy as each data representation within the design must be duplicated to allow both the vector <1110> and the vector <0001> to convey the same information. This forces any implementation to be only 50% efficient in terms of transmitted information. Unfortunately, Shedletsky (1978) does not provide hardware implementation data.

2.5.1.2 Summary of Duplication Codes

Duplication codes are used in areas where the information redundancy is not of great concern and the design time is necessarily short. The main disadvantage with duplication codes is the immediate information redundancy of 100%. However, the advantages with this type of encoding scheme are the ease in which totally self checking (TSC) circuits can be implemented and the conceptually simple nature of the encoding.

2.5.2 Arithmetic Codes

Error correcting codes for arithmetic operations have received considerable attention and can be classified into non-separate and separate codes. Both classes of codes possess many similar properties, but differ significantly in their implementation. The non-separate code considered is the AN code as described by Brown (1960), which is formed when an uncoded operand (N) is multiplied by the check modulus (A) to give a coded operand AN . With this type of coding, the coded word cannot be separated into original data and check bits as the check is integral to the data undergoing the operation. Alternatively, the separate codes considered are the residue code, and the inverse residue code as described by Brown (1960), Garner (1966), Russell (1989), Siewiorek (1992) and Avizienis (1971). These codes are separate since the check bits are generated and appended to the original data to form the transmitted word. In this manner, the data transmitted is of the form IC (where I is the original information word and C is the appended check) and the check bits are easily determined.

2.5.2.1 Non-Separable Arithmetic Codes

The AN codes described by Brown (1960) are a coded form of the operands such that two numbers X and Y are coded as $AX+B$ and $AY+B$. These codes are useful since

the two coded operands can undergo arithmetic operations and the result will differ from the coded output by a constant. Consequently, the coding is preserved by the arithmetic operation. In the case of addition for example, the two coded input operands could be:

$$AX+B + AY+B = A(X+Y)+2B$$

The coded result thus differs by a constant B from the AN coded output of $A(X+Y)+B$. Similar results can also be shown for other arithmetic operations. However, certain requirements must be satisfied by the AN code. Firstly, the minimum Hamming distance between messages must be two for error detection and three for error correction. Secondly the complement of a coded operand must be obtainable by complementing each binary symbol in the coded message. This necessary for subtraction as noted by Brown (1960). However, some logical operations cannot be checked by arithmetic codes and as such are unsuitable for this type of coding. In this case, the operations must be performed on uncoded operands with no error protection as noted in Siewiorek (1992).

2.5.2.2 Separable Arithmetic Codes

The separable arithmetic codes considered are the residue and inverse residue codes as described by Brown (1960), Garner (1966), Russell (1989), Siewiorek (1992) and Avizienis (1971). In the residue-m code, the residue of a data word N is defined as $R(N) = N \bmod m$, with the codeword formed by concatenating N and R(N) to produce NR(N). The result after operation can be checked by comparing the residue of the result with the residue of the inputs undergoing the same operation as shown in Figure 2.3.

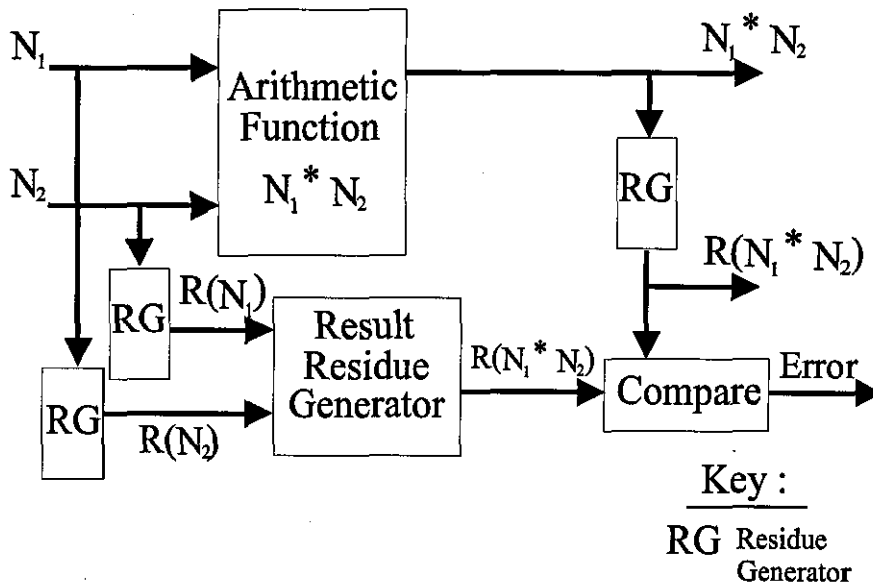


Figure 2.3 - Residue Coding for Arithmetic Functions

If the residues after operation are equal, no error has occurred. One variation of the residue- m code is the inverse residue- m code, where the check quantity, Q , is formed by $Q = m - (N \bmod m)$. The inverse residue code has greater tolerance of repeated-use faults as shown in Avizienis (1971), where a sequence of operations could be performed on faulty hardware before any checking is performed (e.g. iterative circuits such as multipliers and dividers). Unfortunately, the use of residue and inverse residue codes are greatly affected by three factors:

- Generally the complexity of the checking hardware increases sharply as the base (m) increases.
- As the base increases, more bits are needed to represent the residue.
- The error detection ability of the residue increases only slowly as the base increases.

This indicates that the choice of base must be kept as low as possible to ensure a low hardware complexity and the fewest check bits. Russell (1989) states that logical operations can also be coded using residue coding, with only a slight alteration to the basic method. However, Russell (1989) also shows that although the residue code can

be used for coding logical operations, the hardware increase can be up to 35 times greater than for pure arithmetic functions depending upon the function.

2.5.2.3 Summary of Arithmetic Codes

In both the AN and residue codes, the error detection operations can be complex, except when the check base equals $2^a - 1$ (where a is the number of check bits). Thus for binary implementations with check base equal to 3, the number of check bits (a) must be 2 to achieve a low cost solution. Implementations that use the $3N$ (for AN coding), and mod 3 (for residue coding) schemes are termed low cost arithmetic codes as the hardware implementation requirement is minimal as stated in Avizienis (1971). Both the AN code and residue code can protect data as it is operated upon by arithmetic functions, but only the residue code can offer protection when logical functions are used. Consequently, the residue coding scheme seems to offer greater flexibility, especially when the designs contain a large proportion of arithmetic circuitry.

2.5.3 Unidirectional Error Detecting Codes

Unidirectional errors are considered to be one of the most common forms of semiconductor failure as noted in Ashjaee (1977). Consequently, unidirectional error detecting codes have received much attention (e.g. Ashjaee (1977), Burns (1992), Freiman (1962), Berger (1961), Smith (1984), Piestrak (1995), Borden (1982), Dong (1984), Bose (1985) and Blaum (1988)). A unidirectional error is a single or multiple error such that all erroneous bits are either of the '0' \rightarrow '1' or '1' \rightarrow '0' error type within a data word, but not both at the same time. These are also known as asymmetric channel transmission errors. Three types of unidirectional error codes are commonly investigated, these being the unordered codes, the t -bit unidirectional error detecting codes and the burst unidirectional error detecting codes as shown in Figure 2.4.

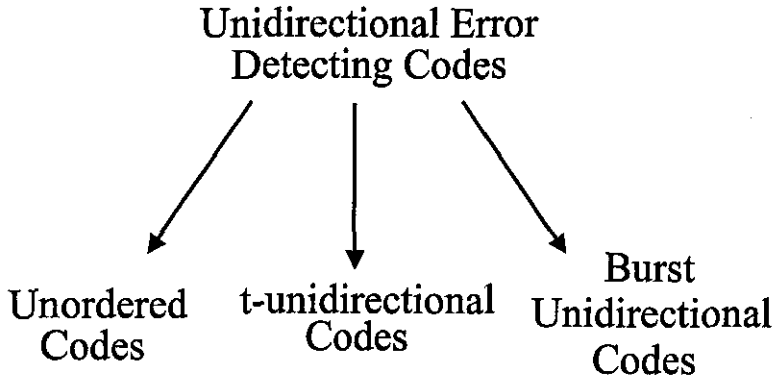


Figure 2.4 - Unidirectional Error Detecting Code Categories

These various coding schemes form the basis of the current work within the field of unidirectional error detection and will be reviewed in more detail.

2.5.3.1 Unordered Codes

Unordered codes are a set of codes that can detect all unidirectional errors in a data word as described in Piestrak (1995). The term unordered is defined by the following conditions:

X covers Y if and only if X contains a '1' everywhere Y contains a '1'.
If neither X **covers** Y nor Y **covers** X, then the binary codewords X and Y are unordered.

Consequently, a binary code C is an unordered code if for every {X,Y} ∈ C, X does not cover Y, and Y does not cover X. For example, the words 1010, 1101 and 0011 form an unordered set, and are capable of detecting unidirectional errors.

2.5.3.1.1 Non-Separable Unordered Codes

The unordered unidirectional error detecting code considered is the m -out-of- n code. The m -out-of- n code (also known as the m/n or constant weight code) is the set of codewords which contain exactly m '1's and $(n-m)$ '0's. This code is non-separable as the original word is not directly identifiable from the codeword and must undergo decoding to obtain the original message. Freiman (1962) and Piestrak (1995) have both offered proofs stating that the m -out-of- n code is optimal when $m = n/2$, as this allows the maximum number of codewords for any code of length n . According to Piestrak (1996) totally self checking circuits can be easily achieved for both combinational and sequential circuits provided that the circuitry is realised inverter free. Also, the detection of a valid codeword within the m -out-of- n set only requires a counter to determine its weight (i.e. the number of '1's in the codeword). This is relatively simple to achieve, but does mean that the counter and the logic function that operate upon the m -out-of- n code are both of crucial importance when designing circuitry. Unfortunately, as the code is non-separable, decoding must take place to obtain the original message.

2.5.3.1.2 Separable Unordered Codes

The separable unordered code considered is the Berger code as originally proposed by Berger (1961) and reviewed again by Ashjaee (1976), Smith (1984) and Burns (1992). The Berger code is separable as the check bits are appended after the original information bits. The check bits are calculated by counting the number of '0's within the information word (I_0) and appending the binary count after the information bits. The Berger code has the property that if all 2^n information codewords are used, it requires the minimum number of check bits to form an unordered code. In this case, the Berger code is *optimal* as defined by Smith (1984). The decoding and checking of information coded as a Berger code by a totally self checking checker has been explored by Burns (1992), Chang (1996), Metra (1996), Ashjaee (1976) and again in

much detail by Piestrak (1995). Currently, the Berger code is most commonly used in applications where a separable code is needed as noted in Burns (1992). There are many totally self checking checker designs for the Berger code that have been explored and are well known (e.g. Burns (1992), Piestrak (1995) and Ashjaee (1976))

2.5.3.2 t-Unidirectional Error Detecting Codes

The data stored within digital systems are usually organised and operated upon in words. Therefore, provided only single hardware failures occur, instead of any number of unidirectional errors occurring as previously described, it is more likely that up to t -unidirectional errors will occur (where t is the word length). Piestrak (1995) describes a t -unidirectional EDC (t -UEDC) as a codeword where no set of t unidirectional errors can transform one codeword into another codeword. The area of t -unidirectional error codes has been the subject of much investigation (e.g. Borden (1982), Dong (1984) and Bose (1985)). As previously, the various codes for detecting t -unidirectional errors can be split into non-separable and separable codes where the check bits are either integral to the transmitted word (non-separable), or directly identifiable (separable).

2.5.3.2.1 Non-Separable t -Unidirectional Error Detecting Codes

The non-separable codes are the Borden codes as originally developed by Borden (1982). These are similar to the m -out-of- n codes discussed previously. The Borden codes are created by using an m -out-of- n code where the weight of n (i.e. the number of '1's or m) is given by:

$$m = \lfloor n/2 \rfloor \bmod (t+1)$$

where n is the length of the codeword and t is the number of unidirectional errors. This code was shown to be optimal in Borden (1982), and stated again in Bose (1985). It should be noted that if $t = 1$, the Borden code becomes identical to the parity code which is separable. This is a unique case of the Borden code as all other versions are non-separable. Self checking checkers for Borden codes have recently been proposed in Piestrak (1995) and have been built using a number of self checking checkers for m -out-of- n codes cascaded. The self checking checkers for Borden codes discussed in Piestrak (1995) are complex and contain a highly irregular internal structure that consequently makes them difficult to design and thus less suitable for complex systems.

2.5.3.2.2 Separable t -Unidirectional Error Detecting Codes

The separable codes capable of detecting t -unidirectional errors are the Bose codes and Dong codes. The separable t -unidirectional error detecting codes were first introduced by Dong (1984) and optimised by Bose (1985). Both codes are similar in that the check bits are appended after the information word, thus forming a separable code. Dong codes are constructed by assuming that any occurring fault can also affect the check bits generated to protect the original information bits. As the check bits themselves are not protected, the Dong codes introduce extra coding to protect the check bits of the original information word. The coding method presented in Dong (1984) uses duplication to achieve check bit protection. In this manner, the entire word is generated in the following manner:

- From the original data word, count the number of '0's (I_0) present and append the binary representation of this number to the original data word. This is a Berger code.
- Perform the bit-by-bit complement of the binary representation of the number of '0's, and append this number after the Berger code.

The resultant Dong code is more expensive than the Berger code in terms of both check bits and the cost of the self checking checkers. Also, their efficiency is slightly lower than the Berger codes for a given number of check bits, although the Dong codes can detect all unidirectional errors of weight less than I_0 (Dong (1984)). The decoding of the received data word consists of a Berger code checker and a two rail checker operating upon the two versions of the received check bits. A detailed explanation of the generation of Dong codes is given in Dong (1984), along with the error detection ability and the method for generating totally self checking checker circuitry.

The second and more optimal separable t-unidirectional error detecting code is the Bose code as described by Bose (1985). Bose codes are separable and require a fixed number of check bits for a particular level of error protection that is independent of the number of information bits. In addition, Piestrak (1995) states that the Bose codes are optimal and self checking checkers are easy to realise. The Bose codes are constructed by counting the number of '0's (I_0) in the information word and taking the binary representation of $I_0 \bmod 2^r$ where r is the number of check bits. As an example, the Bose code for the information word of '00001101' with two check bits is '00001101 01'. The choice of the number of check bits to use is determined by the number of errors requiring detection. Bose (1985) shows that for double error detecting, two check bits are needed, while for triple error detecting, three check bits are needed. The proof of the optimality of the Bose codes is given in Bose (1985), along with the methods for generating totally self checking circuits for checking the received codewords.

2.5.3.3 Burst Unidirectional Error Detecting Codes

Certain faults in semiconductor memories tend to produce bursts of unidirectional errors. Hence, codes capable of detecting bursts of a certain length using a minimum number of check bits are important. To date, no non-separable burst unidirectional error detecting (b-UED) codes have been proposed. However, separable b-UED codes were first proposed by Bose (1986) and improved by Blaum (1988). It has been stated

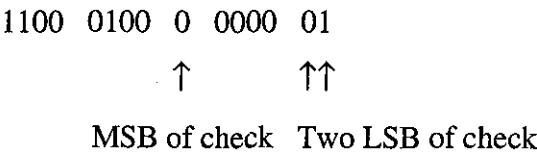
in Piestrak (1995) that if $b(K)$ denotes the maximum length of a burst error detectable by a separable b-UED code with K check bits, then the Bose codes have $b(K) = 2^{K-1}$ for all K . Alternatively, the Blaum codes have $b(K) = 2^{K-1}$ for $K = 2, 3$ and $b(K) > 2^{K-1}$ for $K \geq 4$. Consequently the Blaum codes are more efficient than the Bose codes for larger K . However, both are discussed for completeness.

2.5.3.3.1 Separable Burst Unidirectional Error Detecting Codes

The separable burst unidirectional error detecting codes considered are the Bose codes and the Blaum codes. Bose (1986) originally developed a burst unidirectional error detecting code containing r check bits which could detect any error burst of length up to 2^{r-1} . Any of the t -unidirectional error detecting codes previously discussed are also capable of detecting burst unidirectional errors of length t . However, the Bose code is more efficient than the t -unidirectional error detecting codes in that its error detecting performance stays constant at 2^{r-1} (where r is the number of check bits), while the t -unidirectional error detecting codes suffer from lower error detecting performances for larger numbers of check bits as noted by Bose (1986). Consequently, the Bose code is superior for a larger number of burst errors and larger numbers of check bits. The Bose code checkbits are determined by:

$$\text{Check Bits} = I_0 \pmod{2^r}$$

where I_0 is the zero weight of the information word, and r is the number of check bits. The most significant check bit is then inserted into the information word while the remainder of the check bits are appended to the end of the information word. As an example, if $k = 12$, $r = 3$, and the information symbol is '1100 0100 0000', then the check is given by $9 \bmod 2^3 = 9 \bmod 8 = 1 = 001_2$. Thus the code word will be as follows:

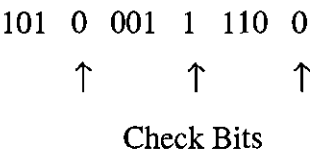


The MSB of the check is placed between the information bits $I_{2^{r-1}+1}$ and $I_{2^{r-1}}$ as shown. Full details of the code construction are given in Bose (1986), along with the design of encoding and decoding circuitry.

The second burst unidirectional error detecting code considered is the Blaum code as presented in Blaum (1988). Blaum (1988) shows that the Blaum code is capable of detecting bursts of errors of length greater than 2^{r-1} when the number of check bits (r) is ≥ 4 . In contrast to this, the previously discussed Bose code is capable of detecting burst errors of length 2^{r-1} when the number of check bits (r) is ≥ 3 . The Blaum code is thus superior to the Bose code and is constructed in the following manner:

$$\text{check number} = I_0 \bmod 2^r$$

where I_0 is the zero weight of the information part and r is the number of check bits. The check bits are found by looking up the check number in tables found in Blaum (1988). The check bits are then inserted into the information word at specific bit positions determined by the minimum of the formula in Blaum. The tables are constructed from mathematical formulae which are beyond the scope of this review but are discussed in detail in Blaum (1988). As an example, consider the case of the information word '101001110', which has weight 4. From the tables in Blaum, the check bits are '010' and the minimum is 4. The constructed code is thus:



This code will detect all burst errors up to length 4. A comparison of the burst error detecting ability of the Blaum and Bose codes for increasing number of check bits is

given in Blaum (1988), where it is shown that the Blaum codes are superior for check bit numbers of ≥ 4 . Methods for constructing encoding and decoding circuits are also given in Blaum (1988), along with totally self checking checker designs.

2.5.3.4 Summary of Unidirectional Error Detecting Codes

The unidirectional error detection codes presented possess a variety of error detection properties, including the ability to detect single, multiple and burst unidirectional errors. The codes usually implemented are the separable codes as these can be easily constructed and checked with simple encoding and decoding of the information bits. This usually involves smaller delays and lower hardware overhead than the generally more complex non-separable codes. The unordered codes considered were the m-out-of-n and Berger codes. It has been stated that the m-out-of-n codes are optimal when $m = n/2$ as this allows the maximum number of codewords, while the Berger codes are optimal in check bits if all codewords are used. However, the m-out-of-n code is non-separable while the Berger code is separable. Consequently, the Berger code is used more due to the difficulty and complexity in encoding and decoding the non-separable m-out-of-n code.

The t-unidirectional error detecting codes considered were the Borden, Dong and Bose codes. Again it was stated that the Borden and Bose codes were more efficient than the Dong codes for a fixed number of unidirectional errors. The Borden codes were shown to be optimal t-unidirectional error detecting codes provided the weight of the codewords was $\lfloor n/2 \rfloor \bmod (t+1)$. Unfortunately, they are non-separable with all the associated problems. The Bose codes were stated to be separable and optimal in check bits.

Finally, the burst unidirectional error detecting codes considered were the Bose and Blaum codes. It was stated that the Blaum codes were more efficient than the Bose codes if the number of check bits is greater than four. Both codes were separable

hence making encoding and decoding easier. To date, no non-separable burst unidirectional error detecting codes are known.

2.5.4 Parity Based Codes

Parity based codes are the simplest coding technique to implement, with the parity bit generated by the EXOR of all the bits in the data word. The parity bit is then appended to the data word to form a code which is capable of detecting single bit errors. Any single bit error is easily detected as the parity will change from odd to even or vice versa. The ability to detect errors can be quantified by the Hamming distance, which is the number of bits by which any two codewords differ. For the simple parity code, the Hamming distance is two, which allows the detection but not correction of single bit errors. Consequently, if two errors occurred, the parity scheme would not be able to detect the errors. In general, any code of Hamming distance $(d+1)$ is capable of detecting d errors, while a code of Hamming distance $(2c+1)$ is capable of detecting and correcting up to c errors. If the Hamming distance is 3, then the code can detect double bit errors and correct single bit errors. The parity code can be generated by several different methods as described below.

2.5.4.1 Bit-per-Word Parity

Bit-per-word parity is a technique where one parity bit is appended to the entire data word. This causes the least hardware overhead as it requires a minimum of redundancy in the information transferred. It also allows a single parity tree to be used for both encoding and decoding of the data provided the data is transmitted in full duplex mode. According to Siewiorek (1992), bit-per-word parity codes can detect all single-bit errors and all errors that involve the corruption of an odd number of bits. A slightly different approach is that of bit-per-byte parity, where an extra bit is appended to each byte of data rather than the entire word as previously discussed. This technique

also detects single or odd numbers of errors in each byte. Another advantage of the bit-per-byte technique is that fewer bits are covered by each parity bit, thus improving the resolution of any diagnostic procedures. This technique also suffers less from the encoding and decoding speed penalties since the parity trees are more compact.

2.5.4.2 Interlaced Parity

Interlaced parity is a technique where i parity bits are appended to the data word. Each parity bit is associated with a group of (b/i) bits (b is the data word length). This is achieved by forming the parity over every b/i_{th} bit, starting in a different bit position for each parity group. Siewiorek (1992) states that interlaced parity covers single bit errors in each group, and multiple errors providing at least one group has an odd number of errors. Siewiorek (1992) also states that if the parity is alternated from group to group, the interlaced parity code also covers a large number of unidirectional errors. As for the bit-per-byte parity technique, the resolution of any diagnosis is to the parity group in error. The speed of error detection and initial encoding is also increased due to the more compact parity tree.

2.5.4.3 Parity Check Matrix

To determine the number of check bits required by any given parity code, the Hamming relationship can be used. This is given by:

$$2^k \geq m + k + 1$$

where m is the number of information bits, and k is the number of check bits required. If, for example, an 8 bit single error correcting code was required, then $m = 8$ and $k = 4$ corresponding to four check bits and 12 bits in total. In a parity check matrix representation, the check bits are inserted into the original message at the power of two bit positions. This has the advantage that comparing the regeneration of the check

bits at the receiver with the received check bits can lead to the correction of a single bit error as described by Russell (1989).

2.5.4.4 Summary of Parity Based Codes

Parity based codes are perhaps the easiest information redundant code to implement as the encoding / decoding consists purely of an EXOR tree and an extra data bit (and storage element if the circuit is sequential). This is easily realised and also provides a reasonably low cost solution to on-line error detection. However, the parity based coding schemes are limited by their ability to detect only an odd number of errors within the group of bits the parity is covering. A trade-off is then apparent between the hardware implementation cost and the level of protection offered by the parity code. Consequently, the parity based codes are best suited to implementations which require a low hardware overhead and can tolerate a lower level of data protection. A totally self checking checker for the parity code consists of a tree of EXOR gates split into two parts such that a two-rail code is generated as described in Khakbaz (1984) and Piestrak (1995).

2.5.5 Summary of Information Redundant Techniques

The technique of information redundancy brings an important benefit, namely the removal of the single point of failure present in most hardware and time redundancy techniques. This can be achieved with the use of totally self checking checkers (see section 2.5.6) which are capable of detecting errors in codewords caused by faults both internal and external to the checker. The various codes available can be used for many different operations, from arithmetic to logic and transmission to unidirectional error detection. Most of the codes presented have been proven to be optimal, and as such represent the best codes available for the type of protection required. The main points of the various information redundancy codes can be seen in Table 2.3.

Coding Scheme	Separable ?	Optimal ?	Error Detection Type	TSCC Available ?
Duplication	Yes	Yes	Any	Yes
Residue	Yes	Low Cost if $m = 3$	Arithmetic	No
Inverse Residue	Yes	Low Cost if $m = 3$	Arithmetic	No
AN Codes	No	Low Cost if $A = 3$	Arithmetic	No
M-out-of-N	No	If $M = N/2$	Single unidirectional	Yes
Berger	Yes	Optimal in Check bits	Single unidirectional	Yes
Borden	No	If Weight is $\lfloor n/2 \rfloor \bmod (t+1)$	t unidirectional	Yes
Bose-Lin	Yes	Optimal in Check bits	t unidirectional	Yes
Dong	Yes	No	t unidirectional	Yes
Bose	Yes	No	Burst unidirectional	Yes
Blaum	Yes	Yes	Burst unidirectional	Yes
Parity	Yes	Yes	Any (Odd number)	Yes

Table 2.3 - Summary of Information Redundancy Techniques

Table 2.3 shows that the various information redundancy codes available have different characteristics. These characteristics include factors such as the optimality of the coding scheme, the separability of the check bits, the types of error the code can

detect and the availability of a totally self checking checker for the code implementation. As the codes presented are intended for use in significantly different environments, it is not possible to categorically state which code is 'the best'. However, the review has shown that the separability of the code is a very useful characteristic, as parallel decoding of the information and check bits can be undertaken. With this in mind, it is possible to highlight the codes that can be implemented easily and consequently those which represent good choices. For arithmetic coding, the residue and inverse residue codes are both separable whereas the AN coding is not. This allows the residue codes to be implemented with less cost than the AN codes. The residue code can also be used to protect logical operations, although the resulting solutions are usually costly in hardware.

For the single unidirectional error detecting codes, the Berger code represents a separable and optimal code which also has the benefit that totally self checking checker circuits are readily available. In contrast to this, the M-out-of-N codes are non-separable and are only optimal for a small set of input codewords, but do also have totally self checking circuits available. Consequently, the Berger codes are usually favoured for single unidirectional error detection.

Of the codes that can detect t-unidirectional errors, the Bose code offers a separable and optimal code, while the Borden code is non-separable and the Dong code is non-optimal. As the Bose code is both optimal in the number of check bits and separable, it represents a good choice for t-unidirectional error detection.

The burst unidirectional error detecting codes available are the Bose and Blaum codes. The Blaum code is separable and optimal, whereas the Bose code is separable but non-optimal. As this is the case, the Blaum code would seem to offer the better choice. However, the generation of the check bits for the Blaum code requires sophisticated algorithms, and is usually implemented as a look-up table in ROM. In contrast, the Bose code check bits are easily generated. A trade-off is then apparent between the required optimality of the code and the difficulty in generating the check bits.

The final code considered is the parity code. The parity code is separable and optimal, but suffers from the disadvantage of only being able to detect an odd number of errors. However, the errors detected can be of any direction (i.e. the code is not constrained to detecting unidirectional errors) and self checking checker circuits are easy to construct.

2.5.6 Totally Self Checking Checkers

Totally Self checking checkers (TSCC) are designed to eliminate the problem of “who checks the checker” in information redundant codes. The idea is to design a checking circuit that also checks itself on-line for any errors. The self checking can be achieved by adhering to three rules:

- Self Testing,
- Fault secure,
- Code Disjoint.

The self testing property is defined as: A circuit is self testing for the fault set F if, for every fault $f \in F$, the circuit produces a non-codeword output for at least one codeword input. This ensures that any fault within the TSCC can be manifested at the circuit output with a codeword input. Secondly, the fault secure property is defined as: A circuit is fault secure for a fault set F if, for every fault $f \in F$, the circuit never produces an incorrect codeword output for any codeword input. Finally, the code disjoint property is defined as: A circuit that maps codeword inputs (non-codeword inputs) onto codeword outputs (non-codeword outputs). Consequently, a TSCC is a circuit which is fault secure, code disjoint and self testing. The design of TSCC circuits is of the utmost importance for information redundancy techniques as this eliminates the single point of failure seen with most time and hardware redundancy techniques. Consequently, most of the information redundancy codes also have complete methods for designing and implementing TSCCs for them (e.g. for m -out-of- n , Berger, Borden, Bose and Blaum codes in Piestrak (1995) and for parity codes in

Gössel (1993)). TSCC circuits for many different types of fault are also described in Cheema (1992), Jha (1993) and Gaitanis (1996).

2.5.7 Partially Self Checking Checkers

Partially self checking checkers (PSCC) are similar in operation to the totally self checking checkers (TSCC) previously discussed. PSCC designs were originally proposed by Wakerly (1973) and essentially divide the input code space into two sections. One section consists of codewords for which the PSCC is totally self checking, while the other section consists of codewords for which the PSCC is not totally self checking. The PSCC is defined to be in 'secure mode' if the input codewords are in the set for which the PSCC is totally self checking, otherwise it is defined to be in 'insecure mode'. If the PSCC operates in 'insecure mode' it is possible that any error may be undetectable. One implementation performed by Wakerly (1973) was the design of an ALU using a PSCC design. Here the ALU operated in 'secure mode' during arithmetic operations, but reverted back to 'insecure mode' during logical operations. With this implementation, residue coding was used for the arithmetic operations while the logical operations were left uncoded. This avoids the difficulty of producing a code capable of checking logical operations.

2.6 Summary

This chapter has reviewed the current state of on-line testing, using the categories of hardware, time and information redundancy. The three categories have been explored and the various advantages and disadvantages of each of the techniques highlighted. As stated in the summary for Hardware redundancy, the hardware redundancy methods alone require a significant amount of additional circuitry to achieve the required on-line error detection. In addition to the increase in circuitry comes an increase in the number of I/O pins, size, power consumption and overall physical weight of the implementation. Finally, the hardware redundant methods also suffer

from the single point of failure, (i.e. the comparator) where any fault would render the on-line test method in-effective. These factors are significant disadvantages and consequently make hardware redundancy alone a less attractive choice for implementing on-line error detection, although duplication with comparison is still used in some areas due to its simplistic nature.

The summary for time redundancy showed that although the majority of the time redundancy techniques require less hardware to implement than the pure hardware redundant methods, they are still quite expensive in terms of hardware overhead. In addition, they also suffer from many of the problems associated with the hardware redundant methods. The most noticeable problems with time redundancy are the increase in hardware, the single point of failure in the comparison stage and a reduction in data throughput. Again the hardware overhead becomes one of the dominant factors that dictate the usefulness of the redundancy techniques in complex designs.

The information redundancy summary stated that on-line test methods require additional bits to be added to the data word requiring protection. This in turn results in an increase in hardware. Information redundancy comes in many different forms, from duplication codes to arithmetic codes and simple parity to unidirectional error detecting codes. These codes give different levels of protection when implemented and different hardware overheads. A trade-off then exists between the required level of on-line error detection and the resultant hardware overhead of the code once implemented. The summaries highlighted the residue codes, parity codes, m-out-of-n codes, Berger codes, Borden codes, Bose codes and Blaum codes as the main information redundancy techniques implemented in contemporary designs, the choice being dependant upon the type of errors expected and the level of protection given by the coding.

2.7 Conclusions

This chapter has highlighted several points which greatly influence the suitability of a redundancy scheme for hardware implementation. These points include increased I/O pin counts, increased complexity, increased power consumption, decreased performance and single points of failure. The required level of on-line test thus needs to be balanced against the implementation penalties to find an acceptable solution. Of the schemes presented, information redundancy offers several benefits that are not obtainable from hardware or time redundancy. These include:

- No single point of failure (provided Totally Self Checking Checkers are used),
- Small increase in I/O pin counts,
- Slight decrease in performance,
- Slight increase in power consumption,
- Slight increase in complexity.

Consequently, information redundancy is an active area of research offering many benefits to the field of on-line test. Many examples of information redundancy being used to aid on-line test have been shown within this chapter, and each implementation brings its own benefits and costs. However, although several approaches employing information redundancy have been shown, there are still many areas where further exploration could be undertaken. The research described in this thesis aims to build upon the benefits of information redundancy, and develop an on-line test methodology using information redundancy as a fundamental component. The information coding scheme used is a parity based dual-rail code under development at Loughborough University. The coding scheme, IFIS (If it Fails, It Stops), uses a parity-per-bit approach to achieve on-line test.

CHAPTER THREE

OVERVIEW OF INVESTIGATIONS

3.1 Objectives of the Chapter

The objectives of this chapter are to select and introduce the investigations carried out within the context of this thesis. More specifically, it includes:

- A statement of research objectives,
- An introduction to the research vehicles employed by these investigations,
- Summaries of the proposed investigations.

3.2 Aims of the Research

This research focuses on an approach to on-line testing under development at Loughborough University. The approach, IFIS (If it Fails, It Stops), is a test methodology that uses dual-rail encoding of data and handshaking between computation elements to achieve on-line test. This research has two aims, namely:

- To identify and assess the encoding schemes, fundamental building blocks and design methodologies for IFIS.
- To evaluate the utility of the concept issues through the construction of a 'realistic' demonstrator IC.

3.2.1 Statement of Research Objectives

This thesis aims to understand the on-line test methodology of IFIS and how to apply it effectively and efficiently to FPGA / ASIC technologies. Four investigations have been proposed to address this area of research and are discussed below.

The first experiment is an investigation into the encoding schemes, processing rules and feedback systems proposed as part of the IFIS methodology. The investigation is a theoretical study with the goal of understanding how the different encoding schemes and feedback systems impact on processing speed and error halting ability.

The second experiment implements the different combinations of encoding schemes, and processing rules into hardware. This is to identify the combination that gives the greatest benefit in terms of processing speed, error halting ability and lowest implementation overhead.

The third experiment improves the complexity overhead of the IFIS methodology. Implementation data on the various IFIS cell re-designs is presented and compared, with the goal of identifying the design that yields the least complexity overhead.

The final experiment is a feasibility study that uses a commercial UART re-designed using the IFIS test methodology. This study gives valuable information as to how well the IFIS methodology scales with design complexity as well as an insight into the different design problems associated with using the IFIS methodology for 'real-life' designs.

3.3 Experimental Vehicles

This section outlines the Finite Impulse Response (FIR) filter and the Universal Asynchronous Receiver Transmitter (UART) which are the key research vehicles of the proposed investigations.

3.3.1 FIR Filter

The FIR filter employed throughout the second experiment was chosen for several reasons, namely:

- The FIR filter is a reasonably simple design, containing only arithmetic functions.
- It is designed to be a vehicle to prove the concept of IFIS, and as such is not required to be complex.
- It has a well-defined architecture that is simple to follow.

These points made the FIR filter a suitable choice, as several of the basic arithmetic blocks (e.g. delay elements, adders and shifters) had already been built and tested for correct operation under the IFIS methodology.

3.3.2 UART

The UART employed throughout the fourth experiment was also chosen for several reasons, namely:

- It is a large design, containing many translation blocks, state machine control units, serial-to-parallel and parallel-to-serial converters. This should allow a comprehensive feasibility study to be undertaken.

- The UART is a commercial available device with well-known characteristics.
- The UART contains several thousand gates, which should give a reasonable estimation as to how well the IFIS methodology scales with increased circuit complexity.

With this in mind the UART seems to be a suitable IC, giving a reasonably large circuit with many different types of electronic designs (e.g. state machines, shifters, counters and parallel-serial converters), but still small enough to be realistically tackled within the given time constraints.

3.3.3 System Environment

The experiments carried out in this thesis were performed using a variety of Computer Aided Design (CAD) synthesis, simulation and test programs. Intergraph's synthesis engine (synovation) was used to perform all of the behavioural VHDL to schematic synthesis, while the simulation engine (DLAB) also from Intergraph was used for simulation of pre-layout and post-layout netlists. The testing of netlists was performed using models of the circuit function written in the 'C' programming language together with VHDL testbenches to apply computer generated test vectors. This particular design route was chosen for several reasons:

- The number of test vectors required to achieve a 'satisfactory' test is usually very large. Consequently, a method of applying an arbitrarily large number of test vectors is required.
- 'Satisfactory' testing is very subjective. Consequently, a method for applying **any** test vector set is required. Using this method, only the test vector file need change.
- No lengthy sequential test vector set is required since the 'C' model and the simulated netlist change 'in lockstep'. Any applied vector will affect both the 'C' model and the netlist, allowing the outputs to still be compared.

- Using a 'C' model together with the design netlist under observation allows a true functional comparison as both the netlist and the 'C' model can continue to transition between any states, and the outputs can still be compared.

With post-layout simulations completed, the netlists were used to configure a Xilinx FPGA. Real-time simulations of the configured Xilinx chip were then performed using LabView™ on a PC based system. With all the experiments, the host systems were a UNIX based SPARCstation and a DOS / Windows based Personal Computer.

3.4 Introduction to Investigations

The first experiment is designed to assess the impact of varying the processing rules, feedback structures and encoding scheme upon the data throughput and error halting ability of circuits designed using the IFIS test methodology. Two encoding schemes are identified which are possible candidates for use within the IFIS methodology. The first encoding scheme uses an 'empty' or non-data carrying state as an intermediate state between data states, while the second encoding scheme uses all possible states to carry data. The various advantages and disadvantages of these two encoding schemes are discussed within the framework of processing speed and error halting ability. Also addressed are the different feedback strategies and processing rules available for the IFIS methodology. Four feedback strategies are identified as possible choices for implementation, and their various advantages and disadvantages are discussed also within the framework of processing speed and error halting ability.

The second experiment is designed to compliment the first experiment and assess the cost / performance impact of the IFIS methodology on real electronic systems. The two encoding schemes and two processing rules previously identified are compared by implementation in hardware. The four different possible combinations are implemented as a digital filter to obtain comparison data. The data is obtained after implementation to both a Xilinx gate array, and the LSI logic ASIC cell library. The

second experiment is a more practical evaluation of the IFIS methodology and allows a conclusion to be drawn as to the encoding scheme and processing rule combination most suitable for use in IFIS systems.

The third experiment involves a re-design of the individual IFIS cells in an attempt to reduce the overhead of implementing the IFIS test methodology. On-Line test methodologies incur a penalty of some kind and that penalty usually involves an increase in circuit complexity. IFIS is no exception, and the third experiment is an investigation into ways of reducing this overhead through a series of individual IFIS cell re-designs. Several choices from chapter two are used as IFIS cell designs, with the goal of identifying the cell design with the least increase in circuit complexity.

The final experiment is the re-engineering of a commercial UART using the IFIS methodology. Many on-line test methodologies presented in the literature are shown only for small circuits of modest gate counts. This does not give any indication of how well the methodology scales with realistic commercial designs containing many thousands of gates. The re-engineering of a commercial UART allows a thorough concept feasibility study to be undertaken. This feasibility study is valuable in many ways. Firstly, the UART contains many different control and translation blocks which need to be implemented in IFIS. This allows a rigorous feasibility study of the methodology on many different types of electronic designs. Secondly, the UART needs to be able to interface with other electronic systems. This requires a study into the practical requirements behind an interfacing scheme for IFIS to non-IFIS systems. Finally, the study will yield comparison data that shows how well the methodology scales with increased complexity of the design.

CHAPTER FOUR

IFIS ON-LINE TEST METHODOLOGY

4.1 Objectives of Chapter

The objectives of this chapter are to understand and assess the factors which impact the effectiveness of the encoding schemes and protocols of the IFIS (If it Fails, It Stops) methodology. In order to achieve these objectives, it is necessary to introduce the IFIS methodology and the different types of encoding schemes, processing rules and feedback structures which operate within.

4.2 Motivation

The IFIS on-line test methodology is based upon the dual-rail encoding of data, designed to detect the logical errors caused by stuck-at, stuck together and transient faults. This is achieved by utilising a parity based dual-rail encoding scheme in conjunction with a set of processing rules which determine when computations should take place. The IFIS encoding schemes and protocols make a novel test methodology which uses information redundancy as a means of achieving on-line test. To perform an efficient implementation of the IFIS encoding scheme, it is necessary to evaluate the various encoding schemes, feedback structures and processing rules. This chapter studies the impact of the IFIS encoding schemes, feedback structures and processing rules on the on-line error detection and data throughput capabilities of systems designed using the IFIS methodology.

4.3 IFIS Encoding Schemes

The IFIS methodology utilises dual-rail encoding of data. This gives the four possible states of :

$$\text{Possible States} = \{00, 01, 10, 11\}$$

The four states can be grouped into the two parity sets:

$$\varnothing_1 = \{00, 11\} \text{ and } \varnothing_2 = \{01, 10\}$$

These parity sets can then be used to determine if an IFIS cell should compute or halt by comparing the input data's parity set against a processing rule. If the input data obeys the constraints of the processing rule, the cell is permitted to compute, otherwise the cell is halted and the previous data value is retained.

4.3.1 IFIS 1 Encoding Scheme

The first encoding scheme, 'IFIS 1', is described by the following state diagram.

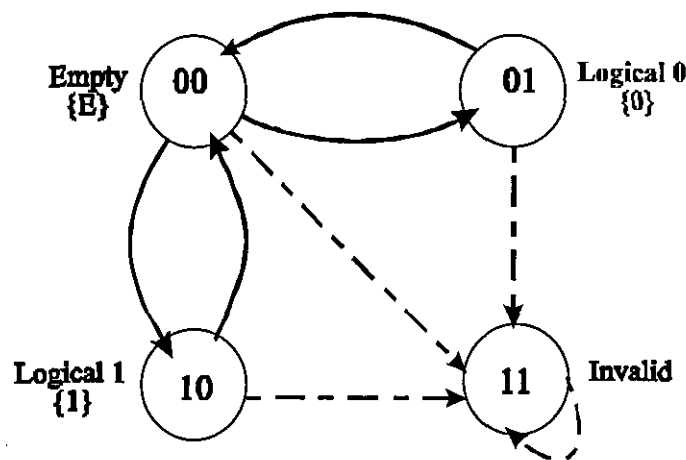


Figure 4.1 - IFIS 1 Encoding Scheme

From Figure 4.1, it can be seen that there are two data states encoded as “10” (for logical ‘1’) and “01” (for logical ‘0’) corresponding to the parity group \emptyset_2 , and two states encoded as “11” (for invalid) and “00” (for empty) corresponding to parity group \emptyset_1 . The states are encoded such that any transition between states is monotonic with the exception of transitions involving the invalid state. This ensures that any desired transition between states within the IFIS 1 encoding scheme causes the parity of the encoding to alternate, and is used to detect errors. Consequently, to transmit a sequence of bits using the IFIS encoding scheme, an alternating parity must be seen for each successive data value. With the IFIS 1 encoding scheme, the empty state must be visited after each data state to maintain the monotonicity of the code and the required parity alternation. Any data stream transmission using the ‘IFIS 1’ encoding scheme must be of the form D,E,D,E,D,E where ‘D’ represents either the logical ‘1’ or logical ‘0’ and the ‘E’ represents the empty state. If the invalid state is entered, or the coding is disrupted (such as a ‘D’ to ‘D’ transition), the parity of the state encoding ceases to alternate and is detected. One disadvantage of the ‘IFIS 1’ encoding scheme is that the data throughput is half that of conventional binary systems as the non-data carrying or ‘empty’ state must be visited after each data state to maintain the monotonicity of the transitions. Another disadvantage is that the IFIS methodology introduces redundancy at the bit level rather than at the byte or word level. This immediately introduces an information redundancy of 100%, which will increase the hardware overhead required to implement the encoding.

4.3.2 IFIS 2 Encoding Scheme

The second encoding scheme under consideration, ‘IFIS 2’, uses all possible states available under the dual-rail encoding as shown in Figure 4.2.

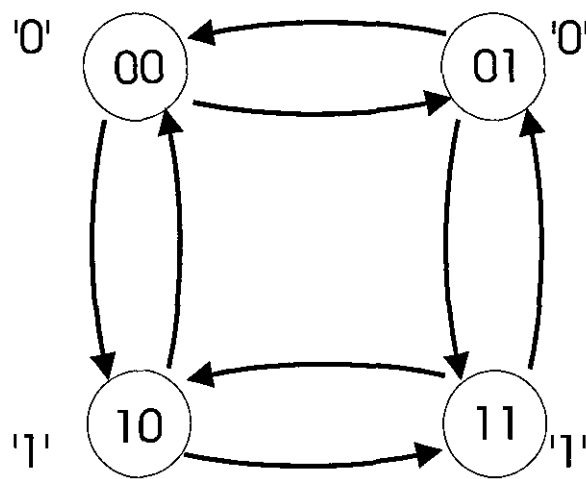


Figure 4.2 - IFIS 2 Encoding Scheme

With this encoding scheme, there are two states representing a logic ‘0’ and two states representing the logic ‘1’. As before, the monotonicity of the transitions must be maintained. The advantage of this encoding scheme is that the empty or non-data carrying state seen with ‘IFIS 1’ is removed and there are no invalid states. This allows the data throughput of ‘IFIS 2’ to be equal to the data throughput of conventional systems. As before, the disadvantage of this type of encoding scheme is that an information redundancy of 100% is introduced. Again this will cause an increase in the hardware overhead required to implement any IFIS systems.

4.4 IFIS Processing Methods

Three methods of allowing each individual IFIS cell to process have been identified. These methods include elastic processing, in-elastic processing and data driven processing. These processing methods control the movement of data through an IFIS system, and are discussed in detail.

4.4.1 Elastic Processing

Elastic processing is the name given to the procedure where cells downstream must finish processing before the cells upstream can begin as shown in Figure 4.3, Figure 4.4 and Figure 4.5.

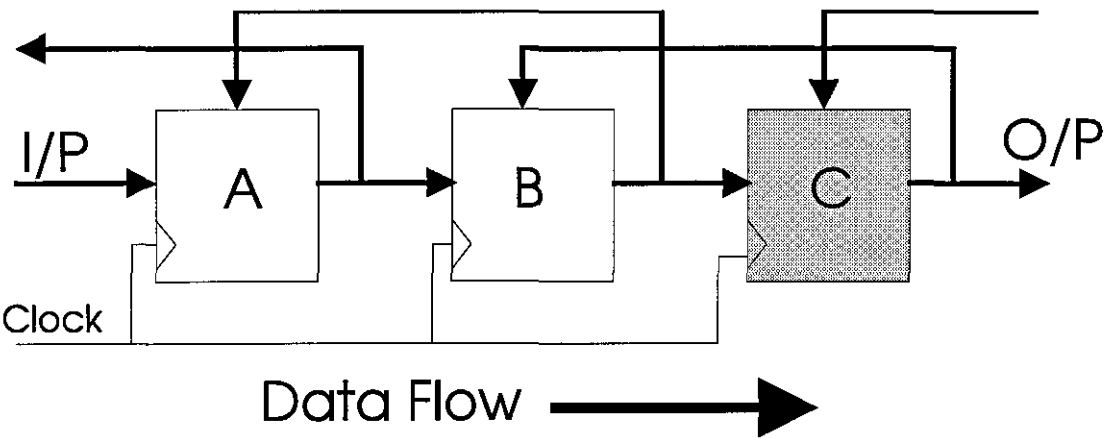


Figure 4.3 - Elastic Processing Stage One

Cell C finishes it's computation (as denoted by the shading) and synchronously signals a completion to cell B which is then allowed to start processing.

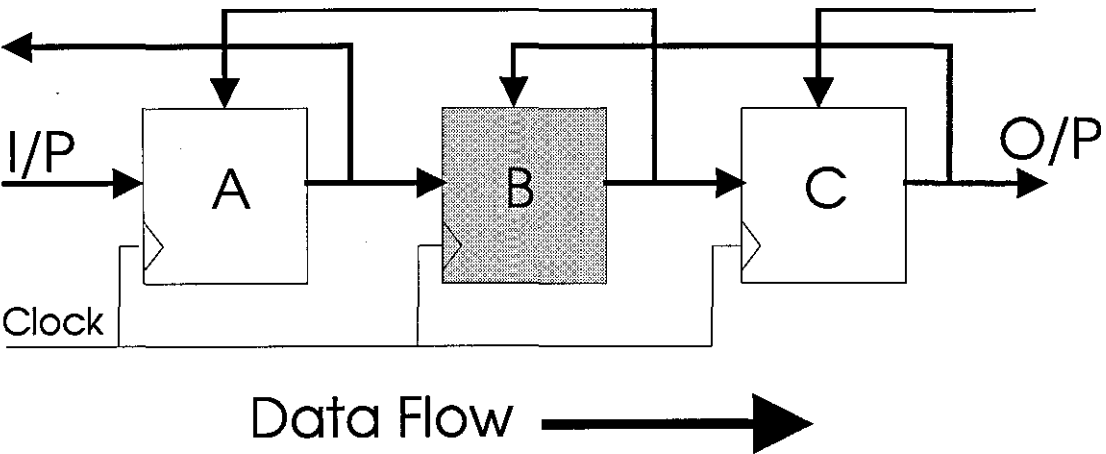


Figure 4.4 - Elastic Processing Stage Two

Cell B then finishes it's computation, and synchronously signals a completion to cell A. Cell A is prevented from processing until the completion signal is received from cell B.

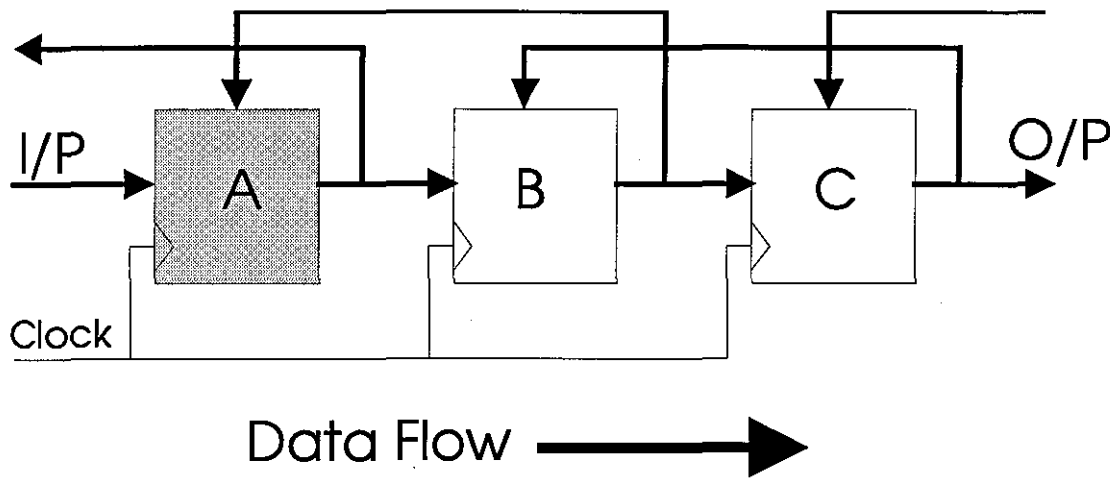


Figure 4.5 - Elastic Processing Stage Three

Finally, cell A is allowed to process as a completion signal has been received from cell B. Processing data in this manner is clock driven as both the new data and the completion signal are synchronised to a clock edge. However, only those cells receiving the required completion signal are permitted to process.

4.4.2 In-Elastic Processing

In-elastic processing is the name given to the procedure where all cells synchronously perform a computation without waiting for a completion signal from downstream cells as shown in Figure 4.6, Figure 4.7 and Figure 4.8.

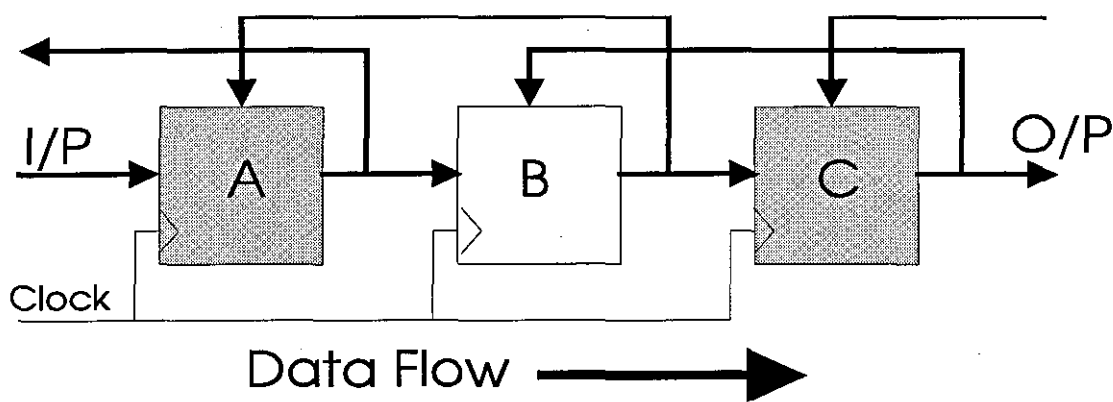


Figure 4.6 - In-elastic Processing Stage One

With in-elastic processing the cells are again clock driven and process data on every clock cycle. Here the shading represents data of the same parity phase.

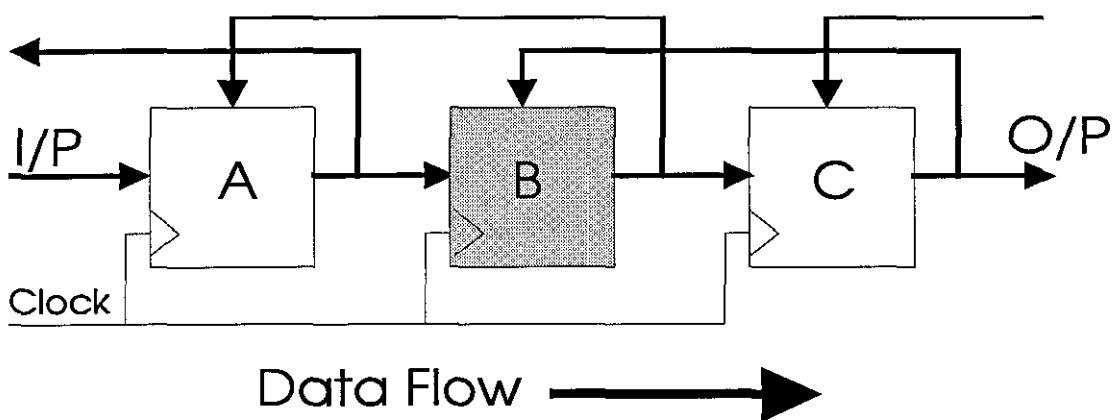


Figure 4.7 - In-elastic Processing Stage Two

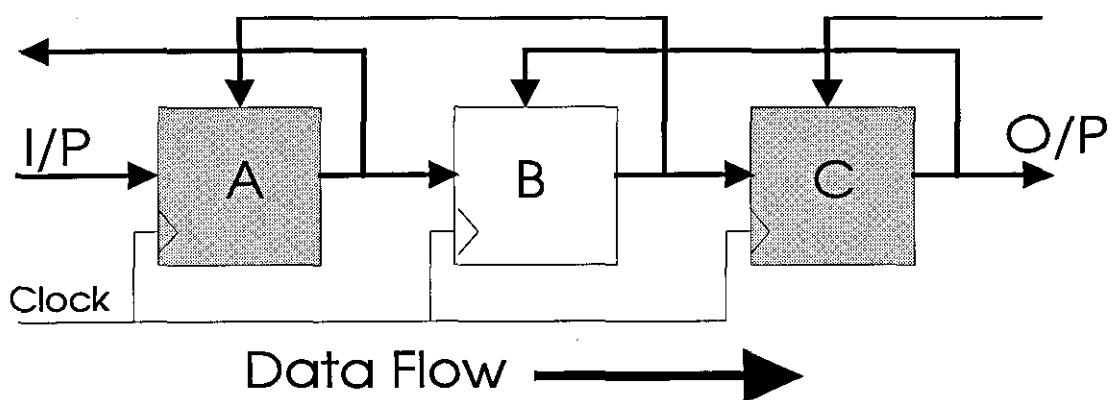


Figure 4.8 - In-elastic Processing Stage Three

The difference between the elastic and in-elastic processing methods is that the in-elastic processing method has all cells processing all of the time, whereas the elastic processing method has a computation wave which propagates upstream governed by the production of a completion signal.

4.4.3 Data Driven Processing

The final processing method is that of data driven processing as shown in Figure 4.9, Figure 4.10 and Figure 4.11. With this system, the computation also operates in a wave like manner, but unlike elastic processing, data driven processing does not produce a completion signal synchronised to a clock edge.

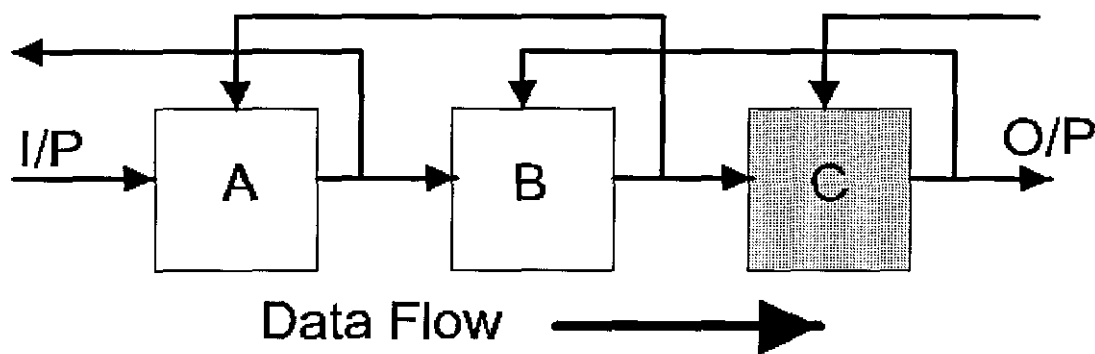


Figure 4.9 - Data Driven Processing Stage One

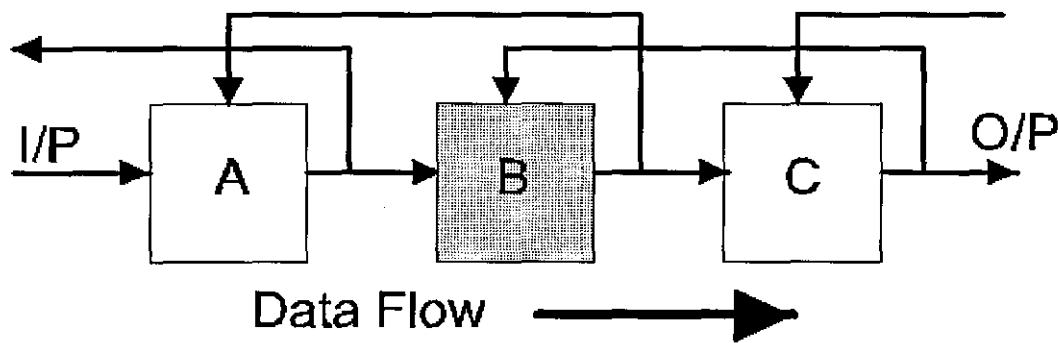


Figure 4.10 - Data Driven Processing Stage Two

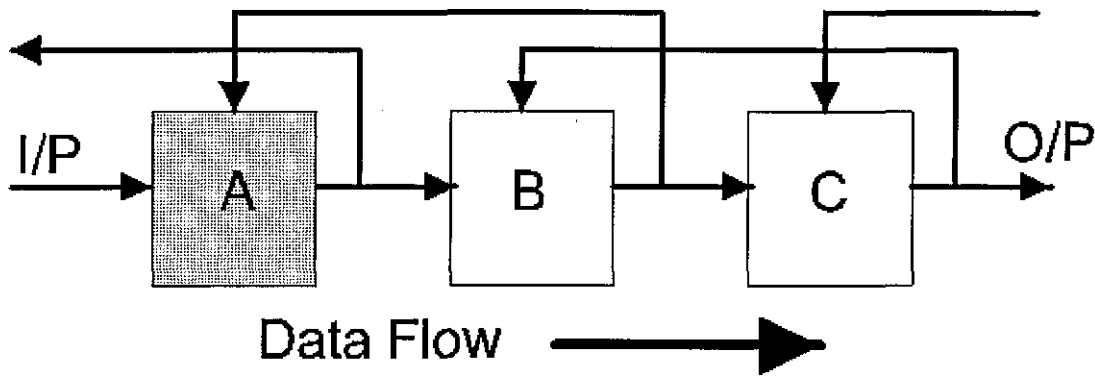


Figure 4.11 - Data Driven Processing Stage Three

Data driven is similar to elastic processing except that the completion signal from the computing cell is generated as soon as the computation completes rather than being synchronised to a clock edge. This allows any system designed in this manner to operate at it's maximum speed while still guaranteeing the correct data output.

With the three methods of processing data introduced, several feedback structures can be investigated.

4.5 IFIS Feedback Structures

The IFIS on-line test methodology also requires the presence of feedbacks to provide communication between cells. If a fault occurs and the encoding scheme is disrupted, the IFIS cell at which the error was first detected is required to stop processing and remain in a halted state. Once this detection has occurred and the faulty cell has halted operation, it is also required that this halting information be transmitted to the surrounding cells so that they may take the appropriate action. The transmission of this information implies that communication between cells must be provided, and

hence some sort of communication structure must be present. Several possible structures immediately present themselves, each with advantages and disadvantages.

4.5.1 Self Feedback

The first structure considered to provide communication about each cells status is self feedback. This is the simplest feedback structure as shown in Figure 4.12 and requires little in the way of extra routing resources as each cell automatically has access to it's own output.

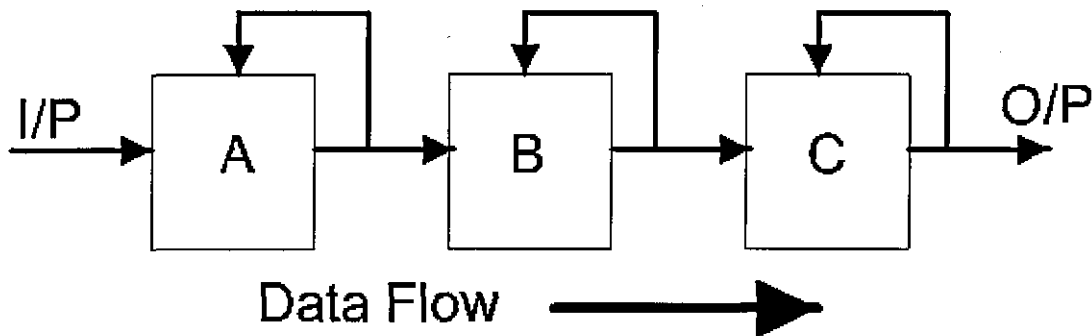


Figure 4.12 - Self Feedback Structure

The elastic and data driven processing methods both require a completion signal from downstream cells before computation can begin. This is not possible with a self feedback structure as no downstream information is ever passed back to upstream cells. The upstream cells thus never know when the downstream cells have finished processing and hence cannot compute. Consequently, self feedback structures cannot support elastic or data driven processing. However, in-elastic processing of data can be supported.

4.5.1.1 In-Elastic Processing

To in-elastically process, all cells must synchronously compute their new data value. This is possible within an IFIS system using the self feedback structure as the individual cells are not waiting for permission to compute from downstream cells. Consequently, to in-elastically process using self feedback, the following processing rule must be obeyed, where \emptyset denotes the data parity :

$$\text{Output} = F(\text{New Data}) \text{ IFF}$$

-- Rule 1

$$\emptyset(\text{New Data}) \neq \emptyset(\text{Self Feedback})$$

By using the self feedback structure, each cell has the previous computational state available for comparison with new data. This allows the cell to determine if the old data and the new data are in opposite parity sets, and hence to process or halt. If the old data and the new data are in opposite parity sets, then computation can continue, otherwise the cell is forced to halt operation and retain the previous (old data) value on its output. If a cell halts because the processing rule is broken, the parity of the output stops alternating. This is then detected by the next downstream cell which also stops processing. The overall effect is a progressive halting of all cells including fanout paths until primary outputs are reached. This type of feedback structure does not affect the data throughput as cells compute in a fully synchronous manner with no completion signal being required from downstream cells. Consequently, the data throughput for self feedback systems using in-elastic processing methods can be one state per clock cycle which is equal to conventional binary systems. However, a disadvantage of this type of feedback structure is that it is possible for a halted system to be restarted with the application of correctly phased data. As the halting propagates forward only, the primary inputs can still accept data. It is then possible for data to be applied to the system which restarts the halted cells, which is a major disadvantage. Another disadvantage of this structure is the forced introduction of delay elements to

synchronise the parity sets of converging data paths. If a cell operates on data arriving from two separate sources, it is possible that the data could be in opposite parity sets. This would cause the processing rule to be violated, and the cell to halt. To prevent this occurrence, delay elements would need to be introduced to synchronise the data parity sets and ensure that any reconvergence did not cause a system halt. Finally, to prevent the processing rule from being violated immediately after a reset, the system must be initialised with a parity phase which alternates along the data path. Otherwise the system would initiate a halt due to a protocol violation as soon as the reset was released.

4.5.2 Successor Feedback

The second feedback solution to provide intercommunication is that of successor feedback as shown in Figure 4.13.

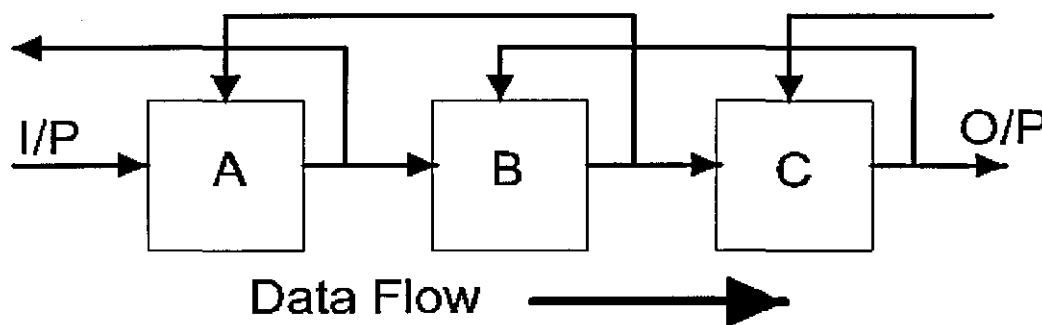


Figure 4.13 - Successor Feedback Structure

This system uses the output of the next downstream cell to provide the information needed to determine if a cell should process. This type of feedback structure requires more in terms of routing resources as the feedbacks originate from the output of a cell which can realistically be placed an arbitrary distance away. This is more costly than

the self feedback structure, but has the advantage that elastic and data driven processing can also be supported.

4.5.2.1 Elastic Processing

Elastic processing is possible with this type of feedback structure as a completion signal is provided from downstream cells. To elastically process with this feedback structure the following processing rule must be obeyed :

$$\text{Output} = F(\text{New Data}) \text{ IFF}$$

-- Rule 1

$$\emptyset(\text{New Data}) \neq \emptyset(\text{Successor Output})$$

Where \emptyset again denotes the parity phase. The feedback signal is compared to the new data arriving at the cell input. If they have opposite parity then computation is allowed to occur, otherwise computation is prevented and the cell halts operation. This halting is then propagated downstream causing all downstream cells including fanout cells to halt. In addition to this, the successor feedback also allows the cell halting to propagate upstream until primary inputs are reached. The advantage of this type of feedback structure is that the halting of cells is propagated bi-directionally towards both primary inputs and primary outputs until the entire IFIS system halts operation. Unfortunately, a disadvantage of this type of feedback structure is that the data throughput is affected by cells awaiting the arrival of a completion signal from the next downstream unit. Consequently, the data throughput of a system employing this type of feedback structure is one state per two clock cycles, as the completion signal can only arrive two clock cycles after the current computation has finished. Another disadvantage of this type of feedback structure is that it may again be necessary to

insert delay elements into the data path to ensure the synchronisation of reconverging parity phases.

4.5.2.2 In-Elastic Processing

In-elastic processing is also possible with the successor feedback structure, with the feedback providing the parity information required to determine if processing should occur or the cell should halt. The changing parity also signals the finish of the computation, but this is not used with the in-elastic processing system as the individual cells are not waiting for the completion signal. With this feedback system, the IFIS cells should process if the following rule is adhered to :

$$\text{Output} = F(\text{New Data}) \text{ IFF}$$

-- Rule 2

$$\emptyset(\text{New Data}) = \emptyset(\text{Successor Feedback})$$

Where \emptyset again denotes the data parity. As before, the halting of an IFIS cell is propagated towards both the primary inputs and primary outputs of the system, including fanout paths, until the entire system halts. Another advantage is that as in-elastic processing does not wait for a completion signal from a downstream unit, the data throughput is not affected by the type of feedback structure used, and consequently is one state per clock cycle. However, as before, care must be taken when designing to ensure that any reconvergent fanout paths do not offer parity sets inconsistent with the processing rule of the reconvergence cell. Again, this may require the insertion of delay elements into the data path. Finally, any system must be initialised with a parity phase which alternates along the data path. Otherwise it would not be possible to apply data of the opposite parity phase to the reset condition.

4.5.2.3 Data Driven Processing

Data driven processing is also possible with the successor feedback structure, as a completion signal is provided to indicate when data can legally be overwritten by upstream cells. As with the elastic processing system, the processing rule is :

$$\text{Output} = F(\text{New Data}) \text{ IFF} \quad \text{-- Rule 1}$$

$$\emptyset(\text{New Data}) \neq \emptyset(\text{Successor Feedback})$$

As before, a disruption of the encoding scheme causes a bi-directional halting to propagate towards both primary inputs and primary outputs. Also, any reconvergent fanout paths must be designed so that consistent parity sets are offered to the processing rule of the reconvergence cell. This also may require the insertion of delay elements into the data path.

4.5.3 Successor Successor Feedback

The third feedback solution uses a feedback structure from two units downstream as shown in Figure 4.14.

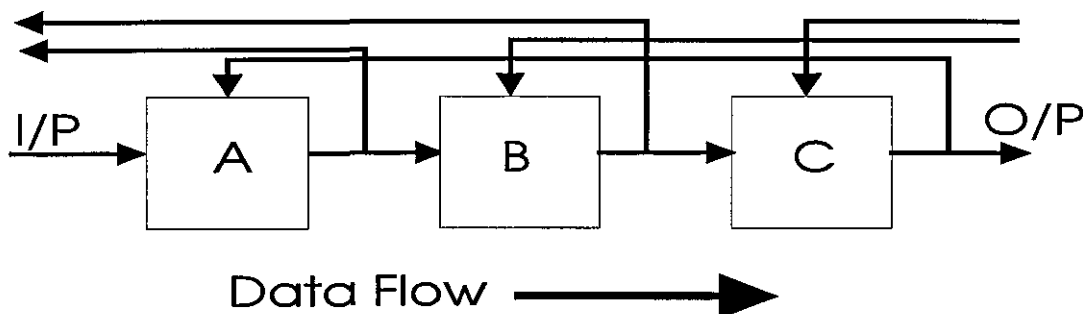


Figure 4.14 - Successor Successor Feedback

This solution is even more costly in terms of routing as the routing path lengths are increased, but also has the advantage that all the methods of processing are supported due to completion signals being available.

4.5.3.1 Elastic Processing

Elastic processing using feedbacks which originate from a position of two cells downstream can be supported as the required completion signal is present. As feedback signals do exist, the bi-directional halting of cells once a fault has been detected does occur. However, a disadvantage of this type of processing with large feedback lengths is the reduction in data throughput caused by the number of clock cycles each cell must wait before the completion signal arrives again. The data throughput drops according to :

$$\text{Max. Data Throughput} = \frac{1}{1 + \text{feedback length}}$$

To successfully process data, the processing rule is :

$$\text{Output} = F(\text{New Data}) \text{ IFF} \quad \text{-- Rule 1}$$

$$\emptyset(\text{New Data}) \neq \emptyset(\text{Successor Output})$$

which is identical to the elastic processing method previously discussed. Again, the insertion of delay elements may be required to synchronise the parity phases of reconverging data.

4.5.3.2 In-Elastic Processing

With in-elastic processing the performance is more favourable as the cells are not waiting for the completion signal to continue processing. Consequently, the data throughput is not affected and remains at one state per clock cycle. However, nothing is gained from using this feedback structure over the successor feedback described previously, and a distinct disadvantage of increasing the feedback distance is the potential routing problem once implemented. The processing rule of each cell using this feedback solution is :

Output = F(New Data) IFF

-- Rule 1

$\emptyset(\text{New Data}) \neq \emptyset(\text{Successor Feedback})$

Once again, systems using this feedback structure and processing rule must be initialised with a parity phase which alternates along the data path. Otherwise it is not possible to apply data of the opposite parity phase to the reset condition.

4.5.3.3 Data Driven Processing

Data driven processing is not advisable with successor successor feedback as data corruption can occur as shown in Figure 4.15, Figure 4.16 and Figure 4.17. Cell A could receive a completion signal from cell C which has finished processing. Cell A would then perform it's own computation, and conceivably overwrite the input data which Cell B had not finished with.

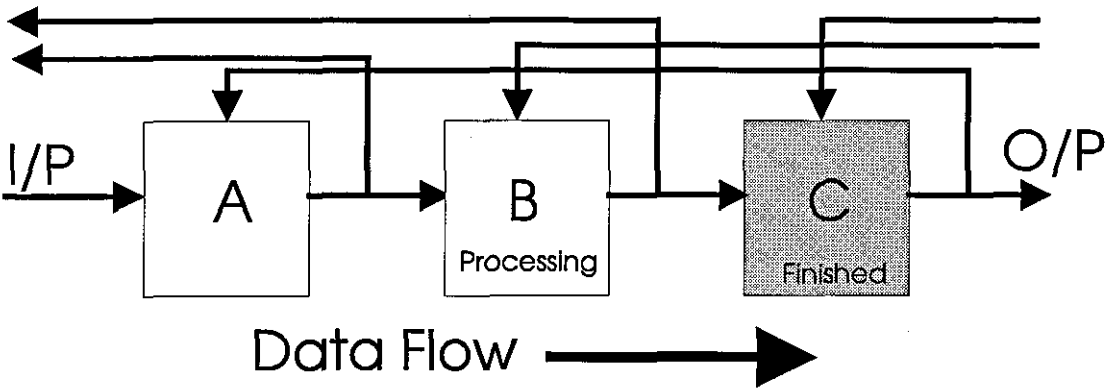


Figure 4.15 - Data Corruption Stage One

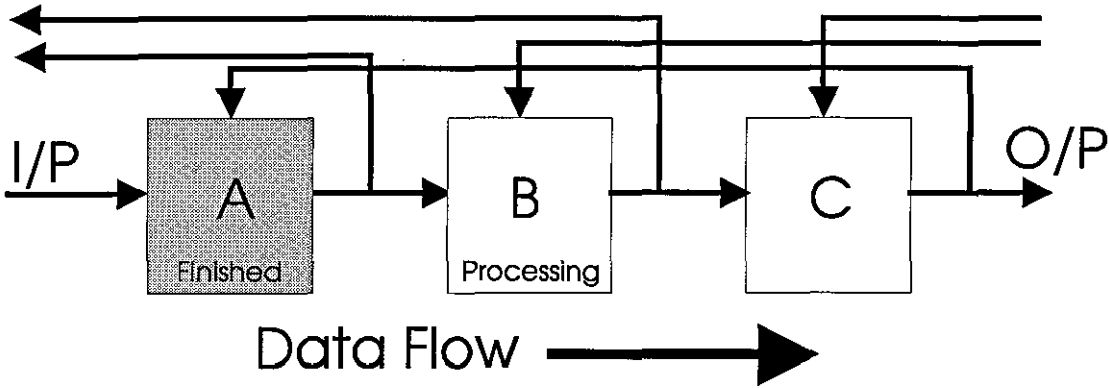


Figure 4.16 - Data Corruption Stage Two

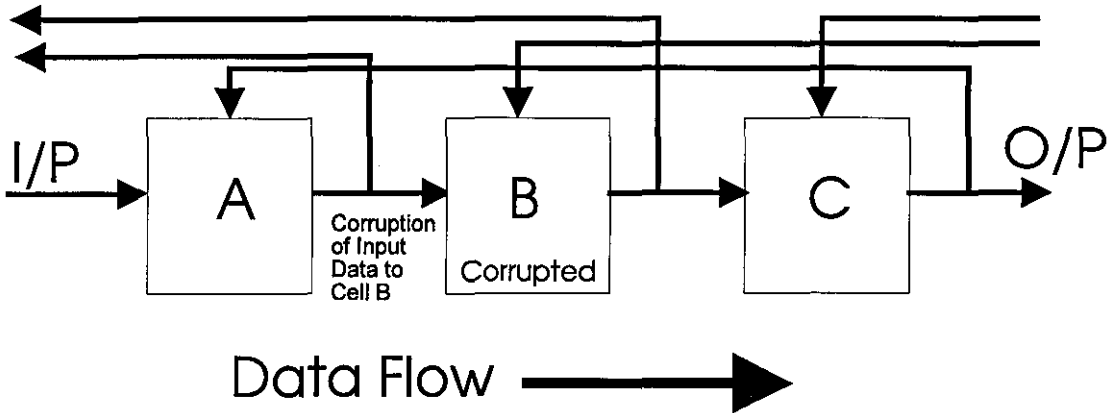


Figure 4.17 - Data Corruption Stage Three

This situation gets increasingly worse as the feedback point is moved further downstream.

Due to the decreased data throughput and increased routing, feedback structures using feedback distances of three cells or greater are not considered further.

4.5.4 Successor and Self Feedback

The final feedback structure considered is a hybrid between the self feedback and successor feedback discussed above, and is shown in Figure 4.18.

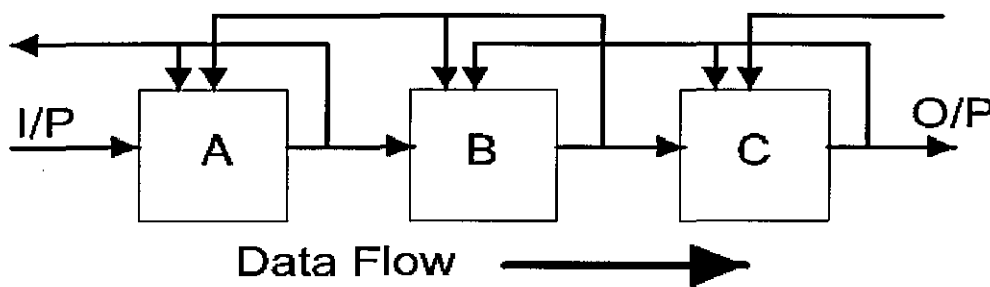


Figure 4.18 - Successor and Self Feedback Structure

The hybrid feedback structure is considered for completion, as both the self feedback and the successor feedback structures have both proven to yield interesting characteristics. This system uses information from both the output of the current cell and the output of the next downstream cell to determine if processing should occur. The twin feedback system increases the routing problem as more feedbacks are introduced to each cell. Fortunately, all processing methods are supported using this feedback structure and are consequently explored.

4.5.4.1 Elastic Processing

Elastic processing is possible with this type of feedback structure, with the feedbacks providing the completion signals required to determine when a cell has finished processing its current data. With this structure, the processing rule is :

$$\text{Output} = F(\text{New Data}) \text{ IFF} \quad \text{-- Rule 3}$$

$$\emptyset(\text{New Data}) \neq \emptyset(\text{Successor Feedback}) \text{ and}$$

$$\emptyset(\text{New Data}) \neq \emptyset(\text{Self Feedback})$$

As with the elastic processing method for successor feedbacks, the halting of cells is bi-directional, and the data throughput is reduced to one state per two clock cycles. Again, the insertion of delay elements into the data path may be required to prevent the system from halting immediately after a reset due to incorrect parity set data being offered to the reconvergence cell.

4.5.4.2 In-Elastic Processing

In-elastic processing is possible using this type of feedback structure, with the feedbacks providing the parity information required to determine if the cells should continue to process. To in-elastically process using self and successor feedback, the processing rule for each cell is :

$$\text{Output} = F(\text{New Data}) \text{ IFF} \quad \text{-- Rule 4}$$

$$\emptyset(\text{New Data}) = \emptyset(\text{Successor Feedback}) \text{ and}$$

$$\emptyset(\text{New Data}) \neq \emptyset(\text{Self Feedback})$$

As before, the halting of any cell is propagated bi-directionally to primary inputs and primary outputs and the data throughput remains at one state per clock cycle. Again, a disadvantage of this type of hybrid structure is that delay elements may need to be inserted into the data path. Otherwise the cell performing the computation on reconvergent data will receive data in conflicting parity sets and halt operation. Also as before, any system using this feedback structure must be initialised to a reset state which contains alternating parity phases along the data path. Otherwise, it is not possible to apply data of the opposite parity to the reset state.

4.5.4.3 Data Driven Processing

Data driven processing is also possible with the self and successor feedback structure, with the feedbacks providing the completion signal required to determine if processing should occur or the cell should halt. With this feedback system, each IFIS cell should process if the following rule is adhered to :

$$\text{Output} = F(\text{New Data}) \text{ IFF} \quad \text{-- Rule 3}$$

$$\emptyset(\text{New Data}) \neq \emptyset(\text{Successor Feedback}) \text{ and}$$

$$\emptyset(\text{New Data}) \neq \emptyset(\text{Self Feedback})$$

As before, the halting of an IFIS cell is propagated towards both the primary inputs and primary outputs of the system, including fanout paths, until the entire system halts. Again, delay elements may be required to maintain the consistency of the parity set information seen by the reconvergence cell.

4.6 Summary of Possible IFIS Systems

From the discussions of the different encoding schemes, processing rules and feedback structures addressed above, it can be seen that each system has advantages and disadvantages which must be evaluated to determine the most efficient and practical solution. Table 4.1, Table 4.2, Table 4.3 and Table 4.4 show the main points highlighted in the above discussions.

Encoding Scheme	Data Throughput	Information Redundancy	Monotonic Transitions ?
IFIS 1	1/2	100%	Yes, if fault free
IFIS 2	1	100%	Yes, if fault free

Table 4.1 - IFIS Encoding Schemes

Table 4.1 shows the two IFIS encoding schemes discussed with their characteristics listed for comparison. Both encoding schemes have monotonic transitions under fault free conditions and both have an information redundancy of 100%. However, it can also be seen that data throughput of 'IFIS 2' is twice that of 'IFIS 1'.

Table 4.2, Table 4.3 and Table 4.4 show the elastic, in-elastic and data driven processing respectively using the different feedback structures.

The four feedback structures discussed each have different characteristics. The first feedback structure discussed was self feedback, which the tables show cannot support either elastic or data driven processing and can also only offer forward halting with the in-elastic processing.

Feedback Structure	Halting Ability	Halt Recoverable	Routing Overhead	Data Throughput
Self	N/A	N/A	N/A	N/A
Successor	bi-directional	NO	Increased	1/2
Greater than Successor	bi-directional	NO	Increased	$1/(1 + \text{feedback length})$
Self and Successor	Bi- Directional	NO	Increased	1/2

Table 4.2 - Elastic Processing of IFIS Cells with Different Feedback Structures

Feedback Structure	Halting Ability	Halt Recoverable	Routing Overhead	Data Throughput
Self	Forward Only	Yes	Negligible	1
Successor	bi-directional	NO	Increased	1
Greater than Successor	bi-directional	NO	Increased	1
Self and Successor	bi-directional	NO	Increased	1

Table 4.3 - In-elastic Processing of IFIS Cells with Different Feedback Structures

The second feedback structure discussed was successor feedback. The tables show that this feedback structure can support all of the data processing methods considered, complete with the non-recoverable, bi-directional halting required by the IFIS methodology.

Feedback Structure	Halting Ability	Halt Recoverable	Routing Overhead	Data Throughput
Self	N/A	N/A	N/A	N/A
Successor	bi-directional	NO	Increased	N/A
Greater than Successor	Possible Data Corruption	Possible Data Corruption	Possible Data Corruption	Possible Data Corruption
Self and Successor	Bi- Directional	NO	Increased	N/A

Table 4.4 - Data Driven Processing of IFIS Cells with Different Feedback Structures

The third feedback structure discussed was greater than successor feedback. The tables show that the elastic and in-elastic processing methods can be supported. However, the elastic processing method suffers from a data throughput which decreases linearly with increased feedback distance, only allowing a cell to process every $P+1$ clock cycles, where P is the feedback length. Also, the data driven processing method has a strong possibility of data corruption.

The final feedback structure discussed was self and successor feedback. The tables show that similar to the successor feedback structure alone, all three methods of processing data can be supported. However, this feedback structure does create an increase in routing above that caused by successor feedback alone. The data throughputs for these structures are equal to the successor feedback structures respectively.

With the basic characteristics of the feedback structures and processing rules examined, some conclusions can now be drawn.

4.7 Conclusions

The comparisons of the IFIS 1 and IFIS 2 encoding schemes offer evidence suggesting that the IFIS 2 encoding scheme should be favoured due to its ability to achieve a data throughput equal to conventional binary systems. However, no circuit implementation data has been presented using these encoding schemes, and consequently the impact of the encoding scheme on hardware overhead is unclear.

Three methods of data processing were identified, namely elastic processing, in-elastic processing and data driven processing. The tables show that the in-elastic processing method is supported by the widest number of feedback structures while the data driven processing method is supported only by the successor feedback structure and self and successor feedback structure. Consequently, the scope of this work is limited to elastic and in-elastic processing as these two methods are best supported. Also, the data driven processing method with different feedback structures has already been investigated in Sutherland (1989), Dean (1991), Dean (1994) and David (1995).

Finally, four feedback structures were identified as possible candidates for use within the IFIS methodology. The self feedback structure cannot support elastic processing or data driven processing. In-elastic processing can be supported, but cannot offer the bi-directional halting required by the IFIS methodology. In addition to this, the halting that is achieved with self feedback can be eliminated if the correct data is applied. As the halting is not permanent, this feedback structure cannot offer the characteristics required by the IFIS methodology which precludes its use.

The successor feedback structure can support all of the identified processing methods, complete with bi-directional halting which cannot be restarted unless a system reset is applied. The only apparent disadvantages perceived with this feedback structure would be the reduction in data throughput if used with elastic processing and the slight increase in routing for the feedbacks themselves.

The successor successor feedback structure can only support elastic and in-elastic processing as data driven processing can result in data corruption. This cannot be allowed to occur and consequently data driven cannot be supported. Both the elastic and in-elastic processing methods can offer the bi-directional halting required by the IFIS methodology. However, the elastic processing method suffers from a data throughput which linearly decreases with the increase in feedback length. Consequently, large feedback lengths are not recommended with elastic processing. In-elastic processing is supported with no effect on data throughput. The only perceived disadvantage of in-elastic processing with this feedback structure is the routing increase from the potentially large feedback lengths.

The self and successor feedback structure is similar to the successor feedback structure in that all processing methods can be supported with the required bi-directional halting. Again, the elastic processing method suffers from a reduction in data throughput to one half that of a similar conventional system. The in-elastic processing method can achieve a data throughput equal to that of a conventional system. The only perceived disadvantage of this feedback structure is the slight increase in routing when compared to the successor feedback structure.

The successor feedback structure has been identified as the most suitable feedback structure for use within IFIS systems. This is due to its ability to support elastic and in-elastic processing and negligible performance impact if used with in-elastic processing. The only disadvantage perceived with this feedback solution is the increase in routing. However, with triple and quad layer metal processes becoming commonplace in contemporary ASIC designs, it is expected that the routing increase can be catered for with little extra cost.

CHAPTER FIVE

IFIS IMPLEMENTATION STUDY

5.1 Objectives of Chapter

The objectives of this chapter are to understand and identify the cost and performance implications of using the IFIS methodology in real electronic systems. In order to achieve this objective, it is necessary to implement the IFIS methodology into real electronic designs. It will then be possible to draw comparison data relating to each implementation of the processing rules and encoding schemes identified in chapter four, and to show realistic figures for the impact of the IFIS methodology on contemporary designs.

5.2 Motivation

The experiment detailed in chapter four highlighted the advantages and disadvantages inherent with each of the encoding schemes, feedback structures and processing rules considered. However, without solid implementation data, it is difficult to identify which combination is the most efficient. This chapter addresses the problem by targeting each combination to the same digital design to obtain data for comparison. This should allow the most efficient combination to be identified.

5.3 IFIS Design Implementation Study

For an implementation study to be undertaken, a suitable vehicle needs to be found which provides a realistic design but which is also reasonably simple to implement. This will allow the IFIS components within the design to be identified and compared, while still providing a range of functional elements which can exercise the methodology. As described in chapter three, the vehicle chosen for this experiment was an FIR filter. This is because the filter contains several shift, addition and multiplication elements, which together with careful choice of the filter coefficients can lead to a simple design. The coefficients were constrained to be 1, 1/2 and 1/4 such that the filter performed an integrating function. This results in a simple design which still contains arithmetic functions needed to exercise the IFIS methodology to a reasonable extent.

5.3.1 FIR Filter Design

The block diagram of a conventional FIR filter is given in Figure 5.1, where the various arithmetic functions can clearly be seen.

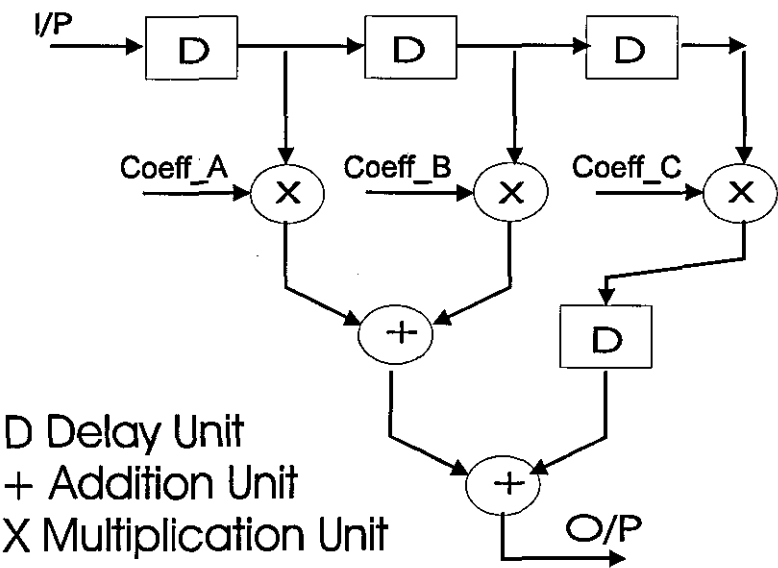


Figure 5.1 - Block Diagram of the Conventional FIR Filter

In contrast to this, the IFIS version of the same filter architecture has some noticeable differences, as shown in Figure 5.2.

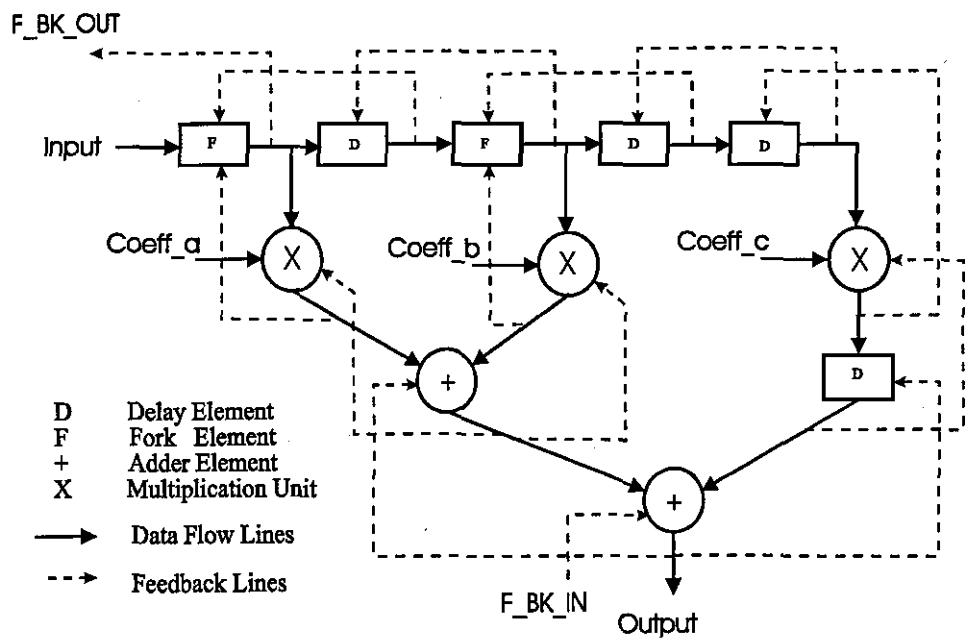


Figure 5.2 - Block Diagram of the IFIS FIR Filter

Figure 5.2 shows that the IFIS version of the FIR filter requires the insertion of several delay elements into the data stream. This is a constraint of the IFIS methodology whereby the system must be reset such that there are alternating parity sets along the data path. If this is not the case, the cells within the system will see combinations of inputs which violate the processing rule and halt operation. Also shown on the diagram is the implementation of the feedback structure chosen from chapter four, with each feedback line originating from the successor cell.

5.4 IFIS FIR Filter Comparisons

The filter architecture described above was used as a basis for the implementation of five designs, one being a conventional FIR filter used for comparison and the other four being the IFIS implementations of the different encoding schemes and processing

rules described in chapter four. These designs are labelled according to the following description :

- Conventional - A single rail conventional FIR filter design
- FIR1i - An in-elastic pipelined FIR filter using the IFIS 1 encoding scheme.
- FIR2i - An in-elastic pipelined FIR filter using the IFIS 2 encoding scheme.
- FIR1e - An elastic pipelined FIR filter using the IFIS 1 encoding scheme.
- FIR2e - An elastic pipelined FIR filter using the IFIS 2 encoding scheme.

Table 5.1 shows general information about the FIR designs which is independent of the target technology.

General Information	Conventional	FIR1i	FIR2i	FIR1e	FIR2e
Latency (Clock Cycles)	4	4	4	24	12
Maximum Throughput (Data States / Clock Cycle)	1/1	1/2	1/1	1/8	1/4
Bi-directional Halting ?	No	Yes	Yes	Yes	Yes

Table 5.1 - General Information about the FIR Filters

As described in chapter four, the choice of processing rule dictates the resultant type of data movement within the FIR filter. Consequently, the processing rule also dictates the resultant maximum data throughput and latency of the design. Table 5.1 shows that the latency of a filter using elastic processing is significantly greater than the corresponding in-elastic filter using the same encoding scheme. This is because the elastic data processing method requires a completion signal from downstream cells before a new computation can occur. Due to the architecture of the FIR filter, feedback loops exist which increase the latency as the completion signal must traverse

the longest feedback loop. The latency is thus affected by the processing method and the architecture of the design.

The data throughput can be affected by two factors. Firstly, the choice of encoding scheme can impact upon the data throughput. If the IFIS-1 encoding scheme is used, the data throughput is immediately reduced to one half of the conventional data throughput. This is due to the non-data carrying state visited after every data state. However, if the IFIS-2 encoding scheme is used, the data throughput is not affected. Secondly, the choice of processing rule can also impact upon the data throughput. The elastic processing method is dependant upon the completion signal arriving from downstream processing cells. The completion signal for elastic processing can only be generated every other clock cycle, and consequently the data throughput is reduced. The in-elastic processing method does not affect the data throughput as a completion signal is generated on every clock cycle. Consequently, the data throughput remains constant. The architecture of the design also affects the data throughput for the elastic processing method. As was seen previously with the latency, a computation stalls waiting for a completion signal from downstream cells. Due to the feedback loops within the FIR filter, the completion signal must traverse the longest feedback loop before the stalled cell can continue processing. This is an architecture dependant constraint which only affects the elastic processing method. The data throughput of the in-elastic processing method is unaffected by the architecture of the design.

Table 5.1 also shows that the detection of a fault by a protocol violation causes a bi-directional halting towards primary inputs and primary outputs which can only be removed if the fault disappears and a reset is applied. Table 5.2 shows some of the most commonly occurring environment and production faults and their impact upon the operation of the IFIS protocol.

Fault Type	Conventional	FIR1i	FIR2i	FIR1e	FIR2e
α -particle (Transient)	Continue	Halt	Halt	Halt	Halt
Stuck-at-0 (Permanent)	Continue	Halt	Halt	Halt	Halt
Stuck-at-1 (Permanent)	Continue	Halt	Halt	Halt	Halt
Bridging (Permanent)	Continue	Halt	Halt	Halt	Halt

Table 5.2 - Fault Types detected by the IFIS Methodology

It can be seen in Table 5.2 that all IFIS encoding schemes protect against stuck-at-0, stuck-at-1, bridging faults and transient faults. This shows the IFIS methodology to be excellent at detecting protocol violations.

LSI lca300k	Conventional	FIR1i	FIR2i	FIR1e	FIR2e
Area (μ)	2022	10098	7384	10038	7506
Maximum Frequency (MHz)	101.6	28.5	49.8	28.6	49.6
Data Throughput (Mbits/Sec)	101	28.5	24.9	7.1	6.2

Table 5.3 - Implementation of the FIR Filters to the lca 300k ASIC cell library

Table 5.3 consists of data drawn from targeting the IFIS FIR filter designs to the LSI logic lca 300k cell library, a triple layer metal 0.6μ process. Table 5.4 shows the same data normalised to the conventional FIR filter.

Relative LSI lca300k comparisons	Conventional	FIR1i	FIR2i	FIR1e	FIR2e
Relative Complexity	1	5	3.6	5	3.7
Relative Maximum Frequency	1	0.3	0.5	0.3	0.5
Relative Data Throughput	1	0.28	0.24	0.07	0.06

Table 5.4 - Relative lca 300k Implementation Data

The data presented in Table 5.3 and Table 5.4 show that the dual-rail nature of the IFIS methodology causes a significant increase in the circuit area of up to 500% for the IFIS-1 encoding scheme, and up to 370% for the IFIS-2 encoding scheme. This can be seen again in Table 5.5 where the same FIR filter designs have been implemented into a Xilinx 4013 gate array.

CLB USAGE (%)	Conventional	FIR1i	FIR2i	FIR1e	FIR2e
I/O	15	30	30	30	30
FUNCTION	14	103	64	104	67
FLIP-FLOP	6	14	14	14	14

Table 5.5 - CLB Implementation of the FIR Filter Designs

Table 5.5 clearly shows a doubling of the required I/O pins for the IFIS designs. This is expected as the IFIS methodology uses information redundancy at the bit level which doubles all signals within a design, including input and output signals. Also shown is an approximate doubling of the required number of storage elements used in the design. Again, this is expected as the number of signals within the design is doubled, thus the number of storage elements is also doubled. Finally, the relative

number of function CLBs required by the IFIS 1 and IFIS 2 encoding schemes are 7.3 (103/14) and 4.6 (64/14) respectively. These figures are both greater than for the ASIC implementations, but still show the expected complexity increase involved in using the IFIS methodology. The difference in area increase seen between the IFIS encoding schemes is due to the IFIS-2 encoding scheme using a saturated state encoding, while the IFIS-1 encoding scheme uses only three of the four possible states. The fourth state in the IFIS 1 encoding scheme is 'illegal' and must be checked for on each transition. Consequently, extra circuitry is required to perform this checking which increases the hardware overhead. This accounts for the difference in gate count and eventual area and CLB usage. Since the area usage and gate counts rise with an IFIS design, the critical path through the design is also going to increase. This causes a decrease in the maximum frequency and also a decrease in data throughput when compared to a conventional implementation. Table 5.4 thus shows the penalty paid for implementing the IFIS methodology, with a significant increase in area, a reduced data throughput and a reduced maximum frequency.

5.5 Conclusions

Chapter five has shown the effect of implementing the IFIS methodology on real electronic systems, and the impact on performance and complexity which results. Two encoding scheme and two feedback structure combinations were implemented in hardware and comparison data drawn to identify the most efficient combination for further investigation. It was shown that the IFIS 2 encoding scheme can achieve twice the data throughput of the IFIS 1 encoding scheme due to the removal of the non-data carrying state. It was also shown that the implementation of the IFIS 1 encoding scheme was 1.58 times larger than the implementation of the IFIS 2 encoding scheme for the same design. It can thus be seen that the IFIS 2 encoding scheme is superior in terms of both data throughput and implementation efficiency. Consequently, the IFIS 2 encoding scheme has been selected for further investigation.

The choice of processing rule is governed by the error halting ability of the system and by the effect on data throughput. It was shown that both elastic and in-elastic methods of processing data achieve the required halting, and consequently the deciding factor is that of data throughput. It was also shown that in-elastic processing can achieve a higher data throughput than elastic processing. Consequently, the in-elastic method of processing data is selected for further investigation. Since it is now known which encoding scheme, processing rule and feedback system to use, larger and more complex systems which utilise the IFIS methodology can be developed.

Finally, the hardware overhead required to implement the IFIS methodology is currently a factor of 3.7 for the IFIS 2 encoding scheme using in-elastic processing. As this is quite significant, chapter six details an investigation into the causes behind this overhead, and looks at possible alternatives in an attempt to decrease the hardware overhead.

CHAPTER SIX

IFIS CELL DESIGN STUDY

6.1 Objectives of Chapter

The objectives of this chapter are to develop efficient circuit design methods for the implementation of IFIS cells. These various methods can then be compared and conclusions drawn as to which cell design method yields the most efficient solution for the implementation of IFIS cells.

6.2 Motivation

The current method of designing IFIS cells leads to a substantial hardware overhead, as was shown in chapter five. The processing rules, feedback structure and encoding scheme of IFIS all contribute to an increase in hardware of up to 370%. In order for the IFIS methodology to become a viable on-line test alternative, this hardware overhead must be reduced. As the IFIS cells are the main contributor to the hardware increase, their redesign should allow a reduction in the hardware overhead required to implement the IFIS methodology. To attempt a cell redesign, it is important to identify the areas within each IFIS cell which contribute most to the increase in hardware, and target the design effort at these points.

6.3 IFIS Cell Internals

Before attempting the redesign of the individual IFIS cells, the behaviour of the IFIS cells must be defined. The behaviour of the IFIS cells can be specified as follows :

- To interact with other IFIS cells according to a strict protocol defined by the IFIS methodology.
- To detect errors within the data or feedbacks supplied by other IFIS cells.
- To flag any errors detected by other IFIS cells by the halting of operation.
- To flag any internal errors when a fault is excited and detected under normal operation.
- To receive data using the IFIS encoding scheme, perform some operation on that data, and then transmit data also using the IFIS encoding scheme.

The high level list of the IFIS cells' behaviour implies that both storage elements and combinational logic are required, and thus the internal design of the IFIS cells can be described with Figure 6.1.

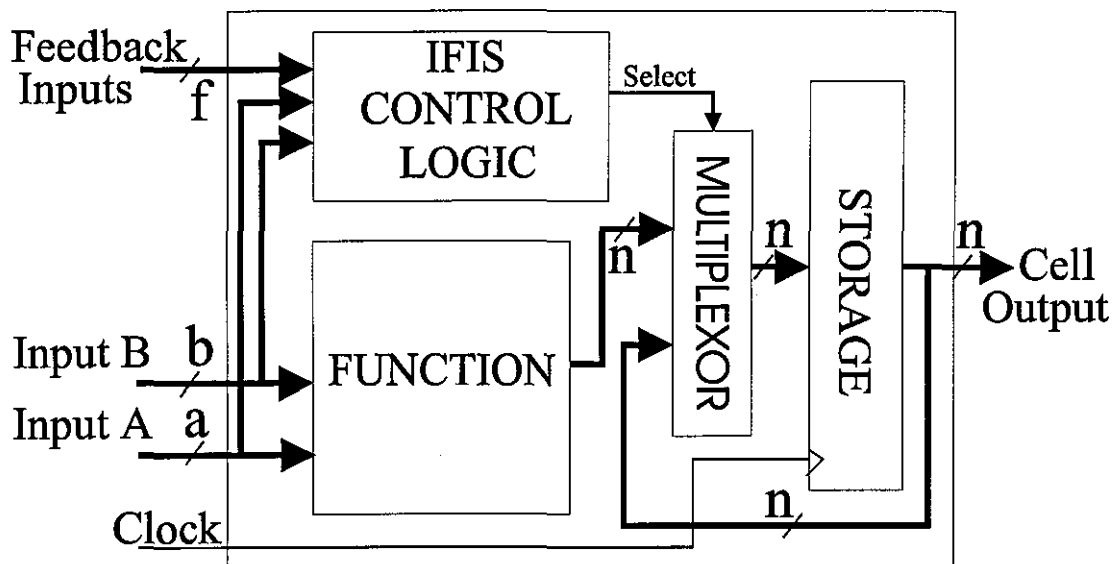


Figure 6.1 - IFIS Cell Internal Block Diagram

One constraint on the design of the IFIS cells is that it contain some form of storage. It is usually considered good design practice to use synchronous design techniques wherever possible as this also aids the off-line testing. Another constraint is that as the functionality of the IFIS cells must match the functionality of their conventional counterparts, an easy method of obtaining IFIS equivalents from conventional functions would be highly advantageous. This would allow the translation from conventional design to IFIS design to be highly automated, and consequently minimise design errors.

6.3.1 IFIS Control Block

The IFIS control block is responsible for the checking of the input data and feedback parity sets against the processing rules defined by the IFIS methodology. The choice of processing rule has been previously discussed in chapter four and chapter five, and it is the IFIS control block which implements this rule. If the input data and the feedback(s) are of the correct parity set according to the processing rule, then computation is permitted, otherwise the last known valid data output is retained. The affect of retaining the last known good data value initiates the halting of the cell and this affect is then propagated to the primary inputs and primary outputs. The in-elastic

processing rule was identified as the most efficient in chapter five, and thus chosen for implementation. Restated, the processing rule is :

$$\text{Output} = F(\text{New Data}) \text{ IFF}$$

$$\emptyset(\text{New Data}) = \emptyset(\text{Successor Feedback})$$

where \emptyset denotes the parity of the data. Consequently, the IFIS control block needs only to ensure the parity sets of the input data and the feedbacks are the same to allow processing. The implementation of this processing rule only depends upon the data arriving at the cell inputs, and the feedback signal(s) from downstream. Consequently, the IFIS control block scales linearly with the increase in the width of the data inputs, and cannot be reduced to any great degree. For a cell which operates on A and B IFIS bit numbers, but also has F IFIS feedback inputs :

$$\text{Number of binary inputs to IFIS control block} = 2(A+B+F)$$

Consequently, the hardware implementation requires $A+B+F$ 2-input EXOR gates to generate the IFIS bit wise parity set, and a comparator which takes $A+B+F$ inputs to determine if the cell should process or halt as shown in Figure 6.2.

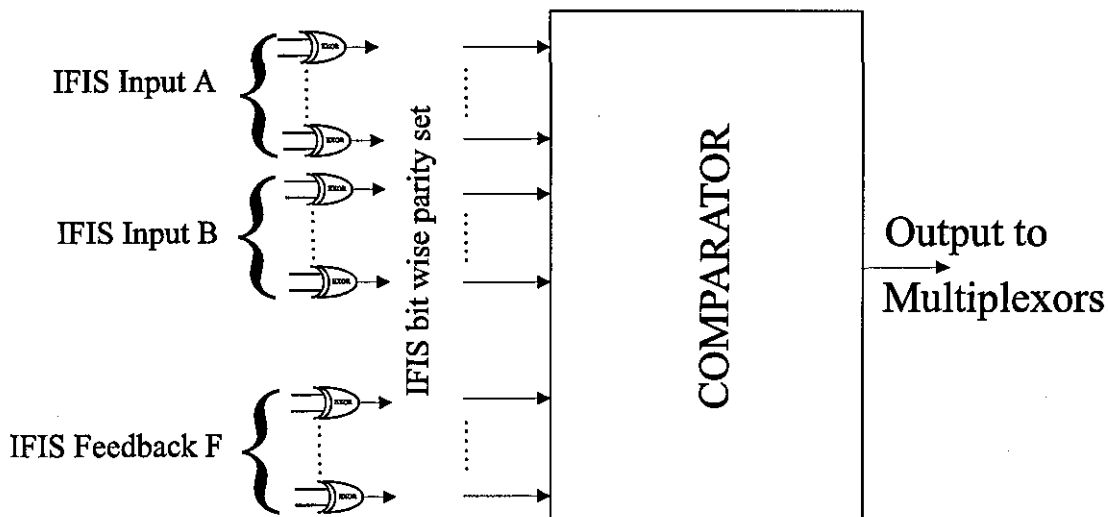


Figure 6.2 - IFIS Data Control Block

This represents the inherent hardware overhead required to implement the IFIS methodology, and cannot be reduced without causing disruption.

6.3.2 Multiplexor Block

The multiplexor block is required to multiplex between the new data computed by the function block and the previous data stored as output from the cell. The multiplexor is controlled by the output of the IFIS control block. If the IFIS control block indicates that the processing rule has been obeyed, the new data from the function block is transferred to the storage elements, otherwise the previous data output is recycled and the cell halting is initiated. As the multiplexor block is directly attached to the output of the function block, the multiplexor block scales linearly with the increase in the output bit width of the function block. However, due to the simplicity of the multiplexor block, which contains as many multiplexors as the cell has outputs, no reduction in hardware can be implemented as this would also cause disruption to the IFIS methodology.

6.3.3 Storage Block

The storage block is present to store the current output of the cell for use by other cells further downstream. The inputs to the storage block are from the multiplexor block, which either recycles the previous output, as in the case of a manifested fault and a halting condition, or allows the output of the functional block to be captured and stored. As with the multiplexor block, the number of elements within the storage block scales linearly with the output bit width of the function block. As before, the structure of this block is quite simplistic, and consequently no optimisation can be performed. As a result, the only block within an IFIS cell which can undergo a redesign in the attempt to reduce the hardware overhead is the functional block.

6.3.4 Functional Block

The functional block contains the operation the IFIS cell is intended to perform upon the input data. No constraint is placed upon the function performed within the function block, and the only requirement placed upon its design is that the output from the function block should be a valid dual-rail IFIS code under fault free conditions. As the function block is currently unconstrained in its implementation, with only the requirement for an IFIS code output, the method of implementation is entirely the choice of the designer. Consequently, the different register transfer level (RTL) coding techniques and logic synthesis tools impact greatly upon the resultant design characteristics. Chapter five showed the hardware overhead for the IFIS methodology to be 3.7 times that of the corresponding conventional binary implementation. As the implementation of the function within the IFIS cell is the only unconstrained block, it is here that the design effort must be concentrated. The re-design can be aided by noting that the individual IFIS bits entering each function block can be separated into two distinct groups. The first group, the 'T' bit group, carries the data (as normal with conventional designs), while the second group, the 'F' bit group, is encoded with the parity set information required by the methodology. As this is true for all IFIS codes, it is possible to use this information to re-design the function block within the IFIS cells. Consequently, several possible solutions for the re-design of the IFIS cell function block present themselves. Each solution has advantages and disadvantages which must be explored before a decision can be reached about the most efficient design method. The design solutions will now be presented and explored.

6.4 Design Solutions

Chapter two explored the current state of on-line testing, and highlighted three distinct redundancy categories aimed at aiding on-line testing. These were hardware redundancy, time redundancy and information redundancy. These three categories form the basis of the investigation into the solutions of the IFIS function block redesign.

6.4.1 Behavioural Synthesis

The first possible solution considered is similar to the original method of the IFIS cell function block design in that it uses behavioural synthesis to obtain the final functional circuit. However, the method of producing the final circuit is constrained to a greater extent than before. This architecture is conceptually the easiest to understand as the design is the result of offering the behavioural description to a circuit synthesiser / optimiser. The IFIS bits are decoding upon input to the function block to determine the parity set. The function then operates on the 'T' input bit lines through a conventional function design. The output is then re-coded in the IFIS manner using the decoded parity set to determine the final output as shown in Figure 6.3.

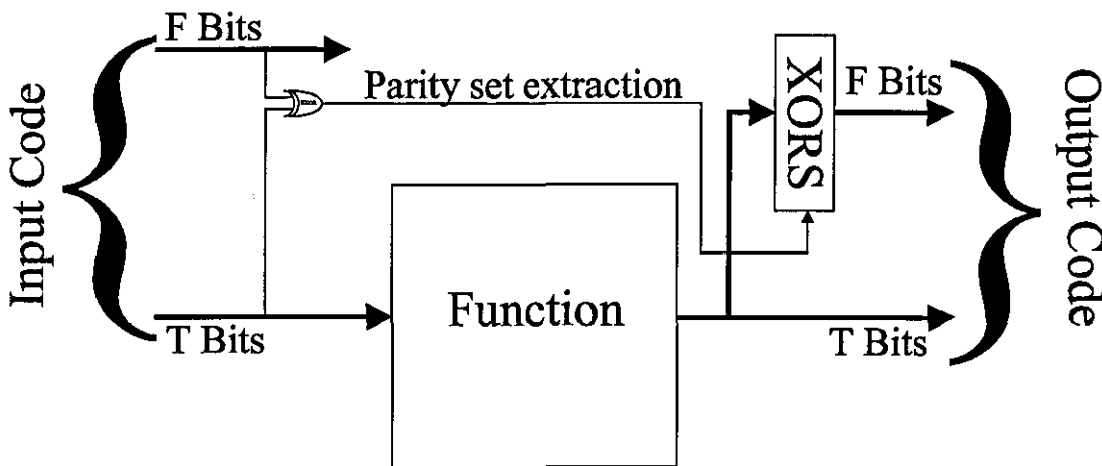


Figure 6.3 - Behavioural Synthesis of an IFIS Cell

The advantages of this technique are :

- The complexity of this technique is only slightly greater than the conventional implementation counterpart due to the EXOR gates required for decoding and re-encoding the IFIS set information.
- The speed penalty incurred is minimal since the only additional delay is that incurred by the EXOR gates, which is generally insignificant when compared to the delay through the binary function.

However, the disadvantage is that :

- A single stuck-at fault within the binary function block will go undetected as the EXOR gates would correct the output and still produce a valid IFIS code.

The complexity overhead for this implementation becomes equal to the conventional implementation area, plus $N+1$ EXOR gates required for the parity correction. (where N is the output bit width from the binary function.)

6.4.2 Arithmetic Coding

The architecture used here is also similar to behavioural synthesis except that the generation of the output codes is achieved by circuitry which is itself self checking as shown in Figure 6.4. This circuitry is based upon the residue checking arithmetic as described in chapter two.

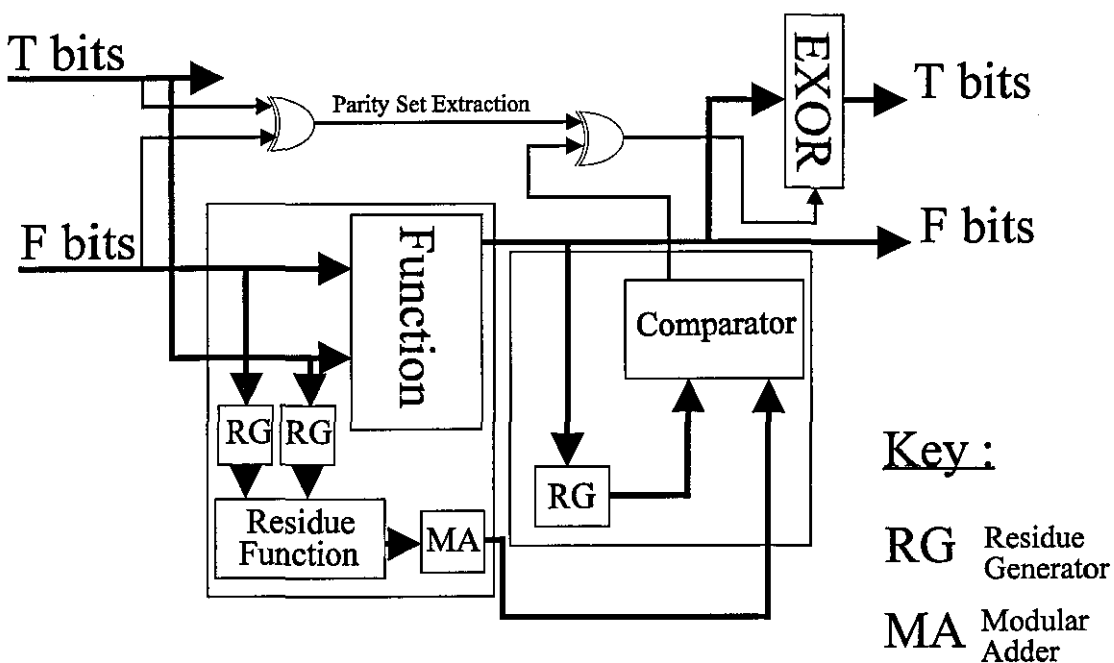


Figure 6.4 - IFIS Cell re-design using Residue Coding

If an incorrect code is detected by the residue checker, a signal is sent to the IFIS output code generation logic which thus causes the output data to assume an incorrect parity. This is then detected as the incorrect parity will cause a system wide halt to occur. The advantages of such a system are :

- The area overhead of the circuit is small in comparison to the binary function, provided the function is non-trivial.
- As a degree of self checking is incorporated into the calculation of the IFIS output code, the output data generation is sensitive to a set of faults which can be detected by the comparator which checks the validity of the residue codes.

The disadvantages of such a system are :

- The low cost residue code employed cannot be guaranteed to cover all the single stuck-at faults which may occur. Although the modulo-3 residue code exhibits a Hamming distance of 2 between codes, this does not guarantee the coverage of all single stuck-at faults in the functional block.
- A more substantial speed penalty is incurred since the generation of the residue code is required to validate the function or otherwise.
- Although it is possible to generate residue codes for Boolean algebra functions, the complexity overhead can be significant in comparison to the area overhead incurred for arithmetic functions.
- This design method does not guarantee fault detection, but relies on the probability of detection inherent in the coding technique.
- The output of the comparator is a single point of failure as a fault on the comparator output line would render the residue code ineffective.

The hardware overhead has been shown in Russell (1989) to be considerable for non-arithmetic functions. As the function block design is aimed at being as generic as possible, this limits the effectiveness of this solution. Finally, the single point of failure from the comparator to the parity generating EXOR gate, could cause the entire system to fail, since a stuck-at good value on this line would be undetectable.

6.4.3 Duality

The hardware redundancy method of duplication was also considered as a solution to the IFIS cell redesign problem. Figure 6.5 shows the block diagram of the considered architecture.

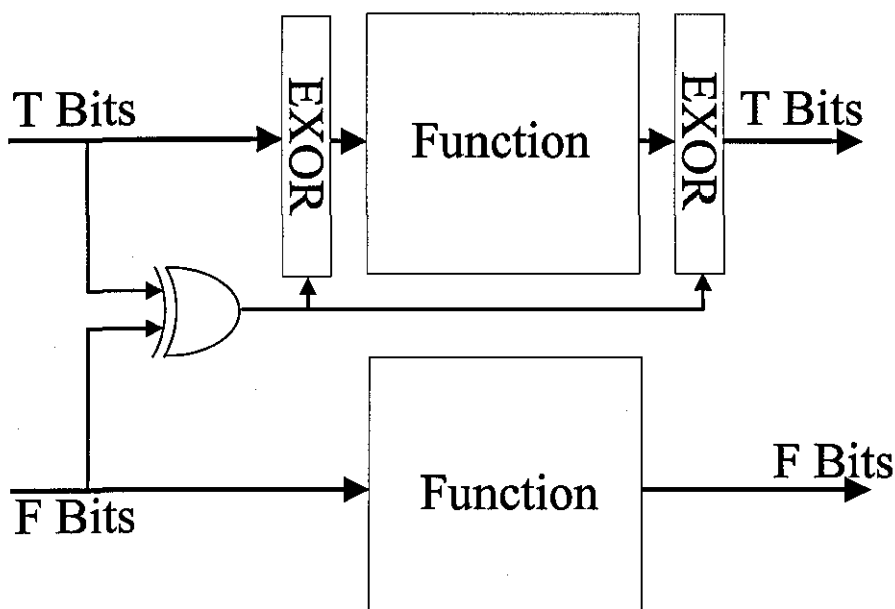


Figure 6.5 - IFIS Cell re-design using Duality

With this architecture both function blocks process identical data. The inputs are decoded by EXOR gates, and the parity extracted from the IFIS data bits. The outputs from the function blocks are then re-encoded by use of the parity data to form the output IFIS code. The advantages of this system are :

- The speed penalty incurred is minimal as both functions operate in parallel. Consequently, the only additional delay is from the single level of EXOR gates which determine the parity set.
- If a single stuck-at fault is manifested and propagated to the output of either function block, the corresponding IFIS code will be incorrect and thus detectable.
- This architecture is generic in that it can be applied to any Boolean or arithmetic function.
- Minimal checking logic is required which avoids complex self-checking checker designs and implementations.

However, the disadvantages of this approach are :

- The hardware overhead for this architecture is immediately double that of the binary equivalent.

Although the hardware overhead is considerable, the removal of the self checking checkers simplifies the design of the IFIS function block considerably and thus may be a valid trade off.

6.4.4 Time Redundancy

The final re-design method uses time redundancy to produce the final IFIS output code. As with the dual architecture discussed before, the IFIS inputs are decoded to ensure that the 'F' bit input data is identical to the 'T' bit input as shown in Figure 6.6.

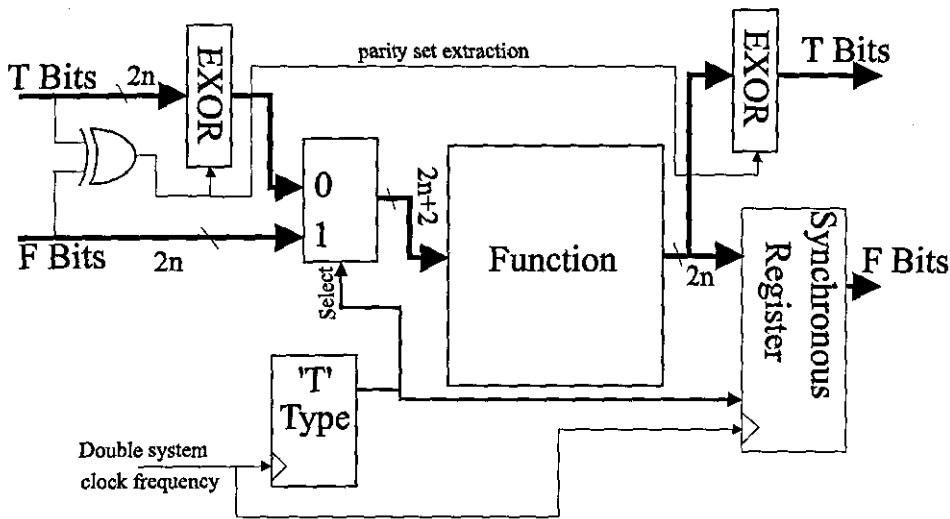


Figure 6.6 - IFIS Cell re-design using Time Redundancy

During the first clock period, the 'F' bits are applied to the function, and the result is stored in a synchronous register. During the second clock period, the shifted 'T' bits are applied to the function, and the result is adjusted according to the input shift and parity set decoded during the first clock period. At the end of the second clock period, both 'T' and 'F' bits are output from the cell. If a single fault has been manifested, there is a high probability that only one of the consecutive operations will be affected as different paths are traversed by the input data. This is similar to recomputation with shifted operands as discussed in chapter two. The advantages of this architecture are :

- The generation of the 'T' and 'F' bits are not closely related, and consequently it is unlikely that any single stuck-at fault would affect more than one of the 'T'/'F' bit pair. Consequently, there is a high probability of any manifested single stuck-at fault being detected by the protocol.
- The complexity of this architecture is not significantly increased in comparison with conventional binary functions as long as the binary function itself is non-trivial.

However, the disadvantages of this architecture are :

- A local clock generator is required to produce the double system clock frequency needed by the internal operation of the architecture.
- Since time redundancy is employed, the data throughput is effectively half that of other methods for the same clock frequency.
- It is possible that there are faults which affect both the output of the function and the output of the shifted inputs in the same manner, thus causing an undetectable fault.
- This technique is suitable primarily for arithmetic operations, and cannot be arbitrarily extended to any Boolean function.

The final disadvantages makes this technique difficult to effectively implement as many of the functions in an IFIS system would be non-arithmetic. Consequently, this technique seems less attractive.

6.5 Summary of Results

A summary of the different function block architectures is given in Table 6.1. The possibilities presented include pure behavioural synthesis, time redundancy, hardware redundancy and the information redundancy scheme of residue coding. From the discussions presented, it can be seen that the dual architecture is the only generic structure which can be easily realised from a conventional binary design. The residue and pipelined architectures must be carefully designed from the initial specification and cannot easily be realised from conventional binary functions. This makes them less generic and thus less attractive approaches. The pipelined architecture also suffers from a reduction in data throughput to one half that of conventional binary data throughputs, although this could be disguised by a higher internal clock frequency. Also, the residue and pipelined architecture are extremely difficult to design for non-arithmetic functions. This severely limits their use as a function block architecture which can be used as a building block for IFIS cell designs. Finally, the behavioural synthesis approach is quite unfavourable since the resultant structure of the IFIS cell is

left entirely to the logic synthesiser and optimiser. This results in an architecture which is difficult to control and reproduce.

	Suitability for Arithmetic	Suitability for non- Arithmetic	Hardware Overhead	Fault Detection	Effect on Circuit Speed
Direct Synthesis	Yes	Yes	100%	Low	None
Residue	Yes	Poor	$\approx 100\% +$ constant	Statistically High	Slight Decrease
Duality	Yes	Yes	$>200\%$	Guaranteed High	None
Pipelined	Yes	Poor	$\approx 100\% +$ constant	Low	Halved

Table 6.1 - Summary of different IFIS function block architectures

With the various IFIS cell architectures explored and comparisons made, some conclusions as to the re-design of the IFIS cell internal blocks can be made.

6.6 Conclusions

The IFIS cell re-design has been attempted with the three redundancy techniques identified in chapter two and the additional technique of behavioural synthesis. The summary highlighted the problems of targeting the pipelined and residue architectures to non-arithmetic functions, and also the reduced throughput inherent in using time redundancy. The in-ability of the pipelined and residue architectures to protect non-arithmetic functions is a major disadvantage, and effectively discounts them both as potential solutions.

The dual architecture is a generic structure capable of being realised directly from conventional binary circuits, with the other architectures requiring extensive design effort from the onset. Although the dual architecture can be directly translated from a conventional counterpart, the increase in hardware of up to 120% is a significant disadvantage. However, the benefits from using the dual architecture are a high fault coverage, a reasonably generic solution and negligible loss in performance.

Finally, the behavioural synthesis architecture is difficult to control and reproduce, with the architecture being highly dependant upon the synthesis tool and the RTL coding style adopted by the designer. This makes the behavioural synthesis architecture difficult to justify as it does not lead to a generic solution. However, the advantages of the behavioural synthesis solution are a small increase in hardware and little or no performance penalty.

The duality solution is consequently selected for further investigation due to it being a generic solution and the negligible performance loss, along with the guaranteed high fault detection.

CHAPTER SEVEN

IFIS FEASIBILITY STUDY

7.1 Objectives of Chapter

The objective of this chapter is to investigate the feasibility of the IFIS concept through the re-engineering of a commercial UART. This will give an indication as to the suitability of the IFIS methodology for commercial adoption, and also an indication as to how the IFIS methodology scales with increased circuit complexity.

7.2 Motivation

The on-line test arena has many different methodologies available for implementation. However, the circuits implemented are generally of very modest complexity with no indication as to how the methodology scales with the increased circuit complexity seen in contemporary ASIC designs. Although a UART only consists of approximately 1,000 NAND gate equivalent cells, it does contain a wide variety of design elements, namely state machines, parallel to serial converters, registers and translation units representative of the components found in ASIC designs. This should allow a thorough concept feasibility study to be undertaken, and also give an indication as to the scalability of the IFIS methodology.

7.3 Design Partitioning of The IFIS UART

The most efficient processing rule and encoding scheme for processing data using the IFIS methodology have been identified in previous chapters. Using the results, the UART can be partitioned and designed accordingly. Figure 7.1 shows the resulting design partitioning of the IFIS UART.

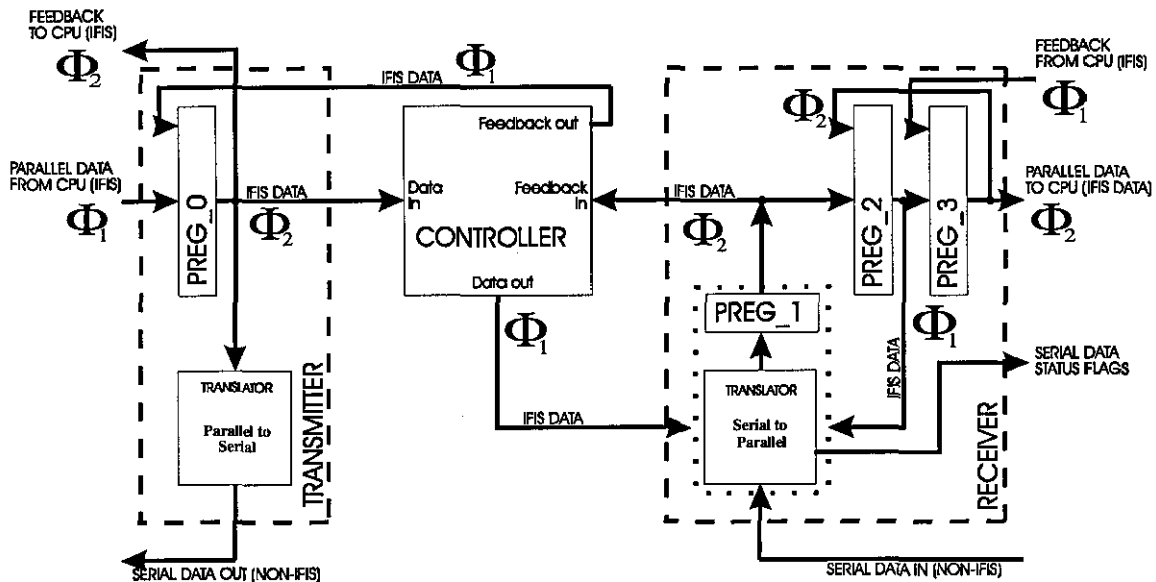


Figure 7.1 - Partitioning of the IFIS UART

Figure 7.1 shows the partitioning of the IFIS UART into the Transmitter, Controller and Receiver blocks, along with the receiver FIFO (First In, First Out) registers and the local serial-to-parallel translation block (for the receiver) and the parallel-to-serial block (for the transmitter). The control block is a state machine responsible for interrupt generation and for co-ordinating the movement of data within the receiver and transmitter blocks. The receiver block is responsible for accepting a conventional serial bit stream, and performing a translation into a parallel IFIS word. Similarly, the transmitter block is responsible for accepting a parallel IFIS word and performing a translation into a conventional serial bit stream ready for transmission. The partitioning of the IFIS UART corresponds directly to the in-elastic processing rule investigated in chapter four. The reset condition of the IFIS UART ensures that each

block is initialised to a parity set which is opposite to the parity set of the surrounding blocks. Each block thus 'sees' inputs of a consistent parity set and is permitted to process. Figure 7.1 also shows the reset parity set information for each block. It can be seen that upon reset each block has consistent parity set inputs as shown by the phases Φ_1 and Φ_2 . This ensures that the parity sets alternate along the data path and is a requirement for the implementation of the in-elastic processing rule.

7.3.1 UART Controller Design

The UART controller is a state machine responsible for the generation of interrupt request signals and for controlling the movement of data through the transmitter and receiver FIFO registers. As the IFIS protocol requires an alternation of the parity set for each consecutive computation, the state machine design must also cater for this. Unfortunately, adherence to the IFIS protocol immediately prevents the use of a state machine which contains self transitions. Any self transition would yield an output whose parity did not alternate across a clock edge and thus violate the protocol. This indicates that any conventional state machine design must undergo a translation to IFIS equivalent before implementation can occur. The translation must map conventional state transitions onto IFIS state transitions which start and finish at states with different parity. Fortunately, when visualised as a state transition diagram, the mapping becomes a reasonably simple graph theory problem whose solution is well known amongst computer scientists. Hadlock (1975), Sahni (1976) and Bolchini (1996) have explored this and similar problems, and state that the solution results in a 'bi-partite' (or two colourable) graph where each conventional state is mapped into two IFIS states, with one state for each parity set. The state transitions can thus be modified to ensure accordance to the IFIS protocol while still retaining the functionality of the original state machine. Duplicating each state to satisfy the IFIS protocol results in a state machine which is twice as large as the conventional counterpart. However, by placing two conventional state machines side-by-side and interconnecting appropriately, the IFIS equivalent state machine can be obtained. The resulting circuit structure is then very similar to the generic approach identified in

chapter six for the design of the IFIS cell internals, and can be realised without extensive design effort.

7.4 UART Verification

As the IFIS methodology is a new on-line test methodology and is targeted to a reasonably complex IC design, it was considered appropriate to verify the operation of both the IFIS methodology and the UART once the FPGA had been configured. Due to hardware/software availability and the independence between the reference model and circuit behaviour, it was decided that 'C' would be the behavioural modelling language used. It was also decided that the 'C' model should produce output data of the same format as that accepted as input by the VHDL testbench. This is because the VHDL testbench was designed so that it accepts files containing expected circuit output values *in addition to circuit stimulus*. This allows the simple comparison of expected values against the values obtained from the circuit outputs during simulation. Furthermore, the file format is the same as that accepted by the LabView/FPGA test rig. The 'C' model can thus be used to provide expected output values for the circuit, whether the stimulus is applied via a VHDL testbench to a design database or to an FPGA via LabView. The 'C' modelling of the UART was partitioned in the same manner as the design. This means that 'C' models of the Controller, Transmitter and Receiver were created. The advantages of this approach are :

- An appropriate model can be used to test each of the major design blocks.
- The models can be developed in parallel.
- The models can be combined, and thus the code re-used during the top-level design validation stage.
- It is possible to generate debugging information at the block level.

The disadvantages of utilising 'C' as a modelling language include a potential to generate unreadable code. In an attempt to reduce this possibility, it was decided to

specify a standard method of modelling. The following standards were adopted for circuit modelling within the UART project :

- The 'make' compilation strategy was used. This ensured that the compiled code was representative of the associated source code.
- Each model was written as a self contained function. This ensured that the model could be combined with other related models at a later date.
- During local code debugging, a local 'socket' was used to test each function. This was a requirement for each block model before simulation.
- A common format header file was used which defines block data structures, vector file input/output structure and debugging flags. This was intended to ease the model combination phase for top-level UART modelling.
- A '#include' file containing all referenced sub-functions was also used to aid readability.

The main advantages of this approach are twofold stemming from the high re-usability of the 'C' source code. Consequently :

- The functions themselves were independently debugged and tested. This guaranteed the resulting model structures were comparable to the circuits which they described.
- The overall development time of the top-level model is kept to a minimum as the block models can be developed in parallel.

However, as 'C' is not a language orientated towards parallel simulation, care must be taken when generating the final top-level UART model. As the Controller block generated the RESET signals for the other blocks, some elements of which were synchronously reset, it was important that the Controller signals were generated before the other blocks were simulated. However, as there was no inter-block communication between the Receiver and Transmitter, the execution order of these blocks was not important. With the modelling of the UART completed, a random IFIS test vector set was applied to the 'C' models and the responses collected. The same test vector sets

were then also applied to the FPGAs containing the IFIS and non-IFIS UART designs, and a comparison performed between the 'C' models expected output and the actual response from the device being tested. This easily led to a pass/fail decision as to the validity of the 'C' model and the device under test. The UART design and the 'C' reference model both received 50,000 test vectors to verify the correctness of both the model and the design, all of which passed successfully. Due to the independent development of the UART and the 'C' model and the successful verification, it can be concluded that the UART design functions correctly and also obeys the IFIS protocol, hence proving the feasibility of designing with IFIS.

7.5 Fault Injection

The design of the IFIS UART also included the means of injecting faults into the controller to show the effect of stuck-at-0 faults within the internal blocks. This was achieved by the insertion of an 'AND' gate into the netlist at pre-defined points. The effect of stuck-at-1 faults could also be shown by replacing each fault 'AND' gate with an 'OR' gate. However, this was not undertaken as the results would be identical to the stuck-at-0 counterparts. Bridging faults were also not injected, as any bridging fault would immediately violate the IFIS protocol thus initiating a halt.

With the method of fault injection established, three fault sites were identified and investigated. These injected fault sites included function block faults, faults on IFIS feedback lines and faults in the IFIS data paths. These fault sites were considered representative as the effect of faults on the interconnection of IFIS cells needs to be established, as does the effect of a fault within an IFIS cell. The data path and feedback faults investigates the effect of interconnection faults, while the function fault investigates the effect of faulty IFIS cells. The fault injections were performed using the configured FPGA and consequently represent real output traces and not simulations. All of the injected faults resulted in the IFIS sections of the UART halting as shown in Figure 7.3, Figure 7.4 and Figure 7.5.

- Points 1 and 2 (Figure 7.2) show the IFIS UART Transmitter and Receiver data busses respectively during fault free operation.

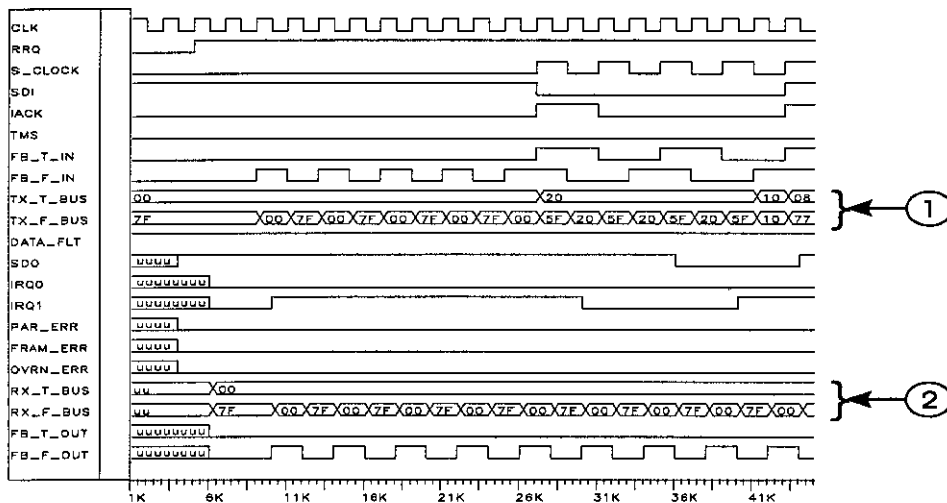


Figure 7.2 - FPGA trace showing fault free operation of the IFIS UART

- Points 3 and 4 (Figure 7.3) show the activation and de-activation of the injected fault on a data path within the controller (DATA_FLT). Point 5 shows the IFIS UART receiver data bus. It should be noted that the even after the injected fault is de-activated (Point 4), the data bus remains halted.

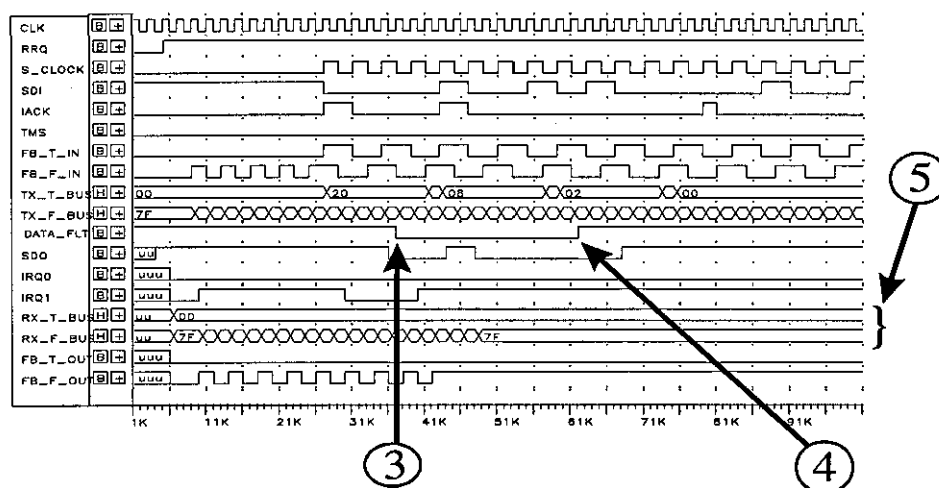


Figure 7.3 - FPGA trace showing IFIS UART halting following fault injection on a controller data path

Figure 7.3 shows that following a fault injection, the RX_T_BUS and RX_F_BUS both halt. As the IFIS protocol is violated, the halting of the RX_T_BUS and RX_F_BUS remains even after the fault is removed (Point 4). In a system designed using the IFIS methodology alone, this would also trigger the halting of all the other components until a system wide halt was established.

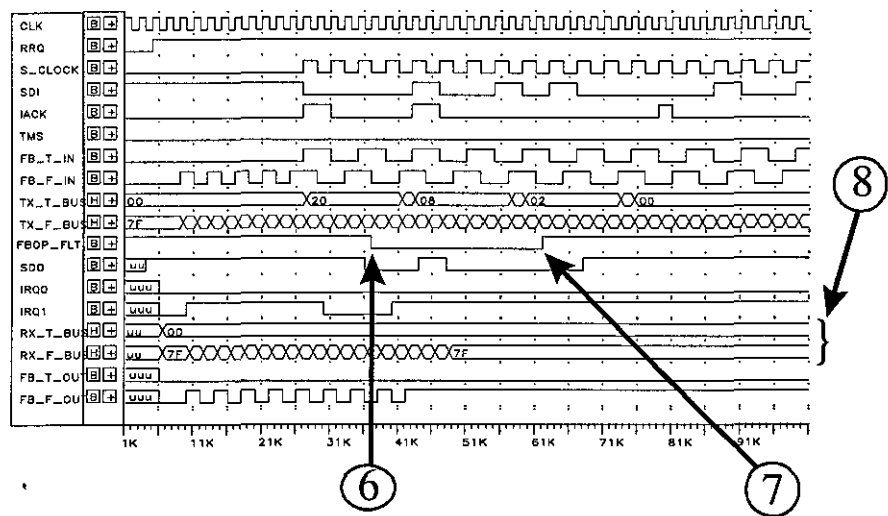


Figure 7.4 - FPGA trace showing halting of the IFIS UART following a controller feedback fault injection

- Points 6 and 7 (Figure 7.4) show the activation and de-activation of the injected fault on a feedback path within the controller (FBOP_FLT). Point 8 shows the IFIS UART receiver data bus. It should again be noted that the even after the injected fault is de-activated (Point 7), the IFIS UART receiver data bus remains halted.

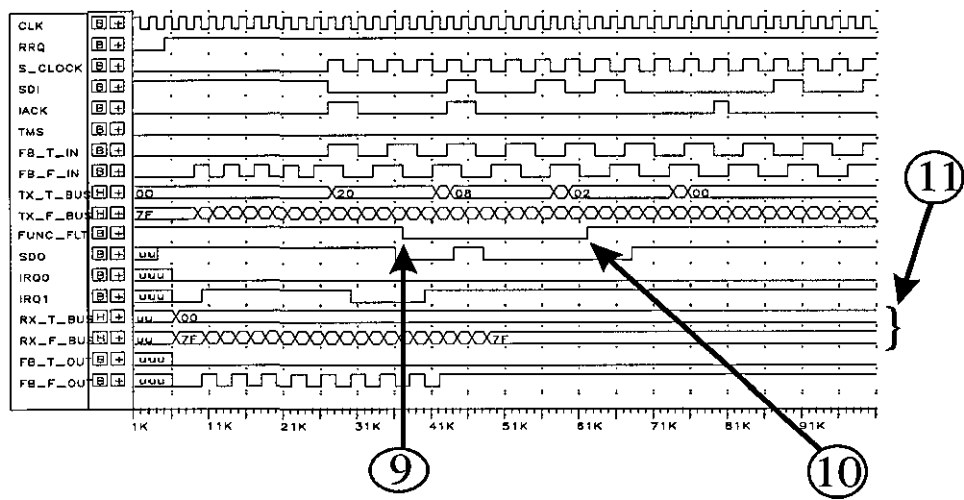


Figure 7.5 - FPGA trace showing halting of the IFIS UART following a fault injected into the controller function block

- Finally, points 9 and 10 (Figure 7.5) show the activation and de-activation of an injected fault within the controller function block (FUNC_FLT). Point 11 again shows the IFIS UART receiver data bus. Once again the data bus remains halted even after the injected fault is de-activated (Point 10).

From the traces, the IFIS methodology can be seen to be fully operational, halting the system immediately a protocol violation is detected. The halting spreads with each clock cycle until all IFIS sections of the system are halted. The system can only be restarted if the fault is removed and a system reset is applied.

7.6 The Impact of The IFIS Methodology

To obtain comparative results, the IFIS UART and conventional UART designs were targeted to the XILINX 4013 FPGA. Comparisons where then drawn from a logic synthesis engine (LOGSYN) and the XILINX place and route toolset (XACT). The controller block of the IFIS UART was implemented purely in IFIS, with the Receiver and Transmitter implemented as a mixture of IFIS and conventional logic. However, all implementation data is shown for completeness. Table 7.1 shows the complexity

overhead inherent with using the IFIS on-line test methodology. The XACT column shows the completely used / partially used CLB (Configurable Logic Block) data, where it can be seen that the hardware overhead is similar to duplication with comparison at 120%.

BLOCK	Methodology	LOGSYN (Gate Count)	XACT fully used CLBs / partially used CLBs
Controller	IFIS	301	101/67
	Conventional	135	47/30
	Ratio	2.23	2.15/2.23
Receiver	IFIS	375	155/106
	Conventional	183	74/49
	Ratio	2.05	2.09/2.16
Transmitter	IFIS	148	72/45
	Conventional	87	37/21
	Ratio	1.70	1.95/2.14

Table 7.1 - Complexity Overhead inherent with using the IFIS Methodology

Due to the IFIS 2 encoding scheme used within the IFIS methodology, the saturated encoding scheme permits the transmission of a new datum with each clock cycle. Consequently, the data throughput remains equivalent to that of the conventional design.

Finally, the XACT place and route toolset allows an estimation of the maximum frequency achievable with the UART designs composite blocks (as each block design was carried out separately). Table 7.2 shows the critical delay through each block, and the corresponding maximum frequency achievable by the design.

UART BLOCK	Methodology	Critical Delay (nS)	Maximum Frequency (MHz)
Controller	IFIS	111	9.01
	Conventional	83	12.05
Receiver	IFIS	106	9.43
	Conventional	106	9.43
Transmitter	IFIS	62	16.13
	Conventional	54	18.50

Table 7.2 - Operational Speed Impact from using the IFIS Methodology

Table 7.2 shows the impact of the IFIS methodology upon the performance of a real electronic design. The table shows that the IFIS implementation of the UART causes an increase in the critical delay, and consequently a decrease in the maximum frequency the design can attain. Table 7.2 also shows that the greatest impact upon performance occurs within the IFIS UART controller, where the use of the IFIS methodology causes a decrease in the maximum frequency by 25%. Similar decreases can be seen for the transmitter although to a lesser extent due to the lower proportion of IFIS components. The critical path through the receiver block is via components which were implemented in conventional binary logic, and consequently no change occurred.

7.7 Scalability of the IFIS Methodology

The design of the IFIS controller requires the doubling of the number of states within the state machine. This can be realised using a generic duplication technique similar to the approach discussed in chapter six, and placing two conventional state machine designs in parallel. Any IFIS state machine thus immediately requires twice the hardware to implement. Consequently, the design of any sequential or combinational circuitry can be translated directly into an IFIS equivalent before implementation. For the UART design, the hardware overhead was seen to be approximately 120%, and

was expected due to the design architecture adopted. For large designs it can be seen that the IFIS technique scales reasonably well, requiring a similar hardware overhead to duplication. However, it should also be noted that for small designs with moderate or little functionality, the IFIS technique does not fare so well. For designs with little functionality at the IFIS cell level, the constant overhead from the IFIS technique can outweigh the hardware required to implement the function. The overhead for the system is thus comprised mainly of IFIS protocol checking hardware and rises accordingly.

7.8 Summary of IFIS UART Case Study

The partitioning of the IFIS UART has been described along with the design of the UART controller and a method of design verification. A means of injecting faults into the final design netlist has also been detailed, and the effect of the injected faults on the operation of the UART has been shown. FPGA traces showing fault free and fault injected UART operation have been shown, proving the halting properties of the IFIS design methodology. The impact of IFIS upon the complexity, maximum frequency and data throughput of a realistic IC design has been explored, and results presented to allow comparisons between IFIS and conventional implementations of the same UART design. The feasibility of designing with the IFIS methodology has been explored and the scalability of the IFIS methodology on 'realistic' ICs has been shown.

The feasibility and scalability of the IFIS methodology have been demonstrated with the design of a commercial UART. The UART design addressed several issues including the design of complex IFIS state machines, the design of IFIS combinational circuitry and the interfacing of IFIS designs with non-IFIS environments. Moreover, the IFIS UART represents one of the most complex designs implemented using an on-line test methodology to date.

7.9 Conclusions

The feasibility of the IFIS methodology has been explored with the design of a 'realistic' contemporary IC, and verified using an independently written 'C' model. The UART and 'C' model both received 50,000 vectors, and the responses were then compared to verify the correctness of operation. As the UART design contains several different design elements commonly seen in contemporary IC design (including state machines, FIFOs, serial-parallel and parallel-serial converters), it can be concluded that the design of systems using the IFIS methodology is feasible, and that both sequential and combinational circuitry can be realised.

The impact of designing with the IFIS methodology has been shown for the commercial UART design, and leads to a hardware overhead of 120% which is comparable with duplication. For any design containing state machines which require a translation to IFIS equivalent, a similar overhead will be created. This can also be seen for combinational circuitry where the dual architecture adopted in chapter six leads to a hardware overhead which is again comparable to duplication. The scalability of the IFIS methodology has been shown to favour designs with large amounts of functionality as this offsets the constant IFIS overhead accordingly. However, with increased complexity becoming common-place, the overhead of IFIS is expected to be negligible in comparison to the functions performed in contemporary ASIC designs. The IFIS methodology could thus offer an attractive alternative, provided the duplication overhead can be tolerated.

CHAPTER EIGHT

CONCLUSIONS

8.1 Objectives of Chapter

The objective of this chapter is to draw conclusions about the IFIS on-line test methodology from the experiments performed in previous chapters. This will allow the suitability of the IFIS methodology for use within the on-line test arena to be appraised, complete with inherent benefits and penalties.

8.2 Review Of Thesis Objectives

The main objectives of this thesis were stated in chapter three. To re-iterate, the objectives were :

To understand how the different encoding schemes and feedback systems impact on data throughput and error halting ability of systems using the IFIS methodology.

To identify the combination of encoding scheme, feedback structure and processing rule which gave the greatest benefit in terms of data throughput, error halting ability and lowest implementation overhead.

To evaluate IFIS cell designs and identify a generic solution which yields the least complexity overhead.

To evaluate the feasibility of the IFIS methodology using a substantial commercial design re-engineered using the IFIS methodology.

8.3 Experimental Investigations

The objectives were satisfied by four experiments. The first experiment was an investigation into the encoding schemes, processing rules and feedback systems incorporated within the IFIS methodology. The investigation was a study into how the different encoding schemes, processing rules and feedback systems impacted on data throughput and error halting ability of designs using the IFIS methodology.

The second experiment implemented the different combinations of encoding schemes and processing rules into hardware. This helped identify the combination that gave the greatest benefit in terms of data throughput, error halting ability and lowest implementation cost. This experiment also used data obtained from the first experiment to choose the most efficient combination.

The third experiment used several different implementation schemes identified in chapter two to discover a more efficient solution for designing IFIS cells. Implementation data from the various re-designs of IFIS cells was compared to help identify a more efficient and more generic design solution.

The final experiment was a feasibility study using a commercial UART re-engineered using the IFIS methodology. This study drew together the results from the previous experimental chapters to aid the UART design. The experiment gave valuable information as to how the IFIS methodology scales with design complexity, and an insight into the different design problems associated with using the IFIS methodology for 'real-life' designs.

8.4 Main Conclusions

Chapter four explored several different processing rules, feedback structures and encoding schemes which were all candidates for implementation using the IFIS methodology. These schemes were evaluated using the metrics of halting ability and data throughput. The conclusion was reached that the successor feedback structure was superior in terms of data throughput, while still achieving the required halting upon the detection of a protocol violation. A preliminary conclusion was also reached that the IFIS 2 encoding scheme was likely to provide the most efficient hardware implementation.

Chapter five detailed the implementation of an FIR filter using the successor feedback structure identified in chapter four. This allowed an investigation into the IFIS encoding schemes and processing rule combinations which control the movement of data through the system. The chapter concluded that the in-elastic method of data processing together with the IFIS 2 encoding scheme was the most efficient combination as this required the least hardware to implement. However, although the IFIS 2 encoding scheme and in-elastic processing rule together with the successor feedback structure represent an efficient means of implementing the IFIS methodology, the hardware overhead was seen to be a substantial 370%.

Chapter six discussed the reasons why the implementation of the IFIS methodology resulted in such a large hardware overhead, and detailed an experiment designed to identify ways of reducing the hardware overhead through the efficient redesign of IFIS cells. The chapter concluded that the most efficient design solution for the IFIS cells is that of duality. This can be easily realised from conventional binary implementations, can be used for arithmetic and non-arithmetic functions, and also results in negligible performance degradation.

Chapter seven discussed the design of a commercial UART re-engineered using the IFIS methodology as a feasibility study. The cell design identified in chapter six was employed, along with the feedback structure, processing rules and encoding schemes

identified in chapter four and chapter five. The UART was successfully tested using an FPGA and a PC running LabView as proof of functionality. The chapter concluded that the design of systems using the IFIS methodology is feasible. However, the implementation overhead could prove to be prohibitive for applications which contain little or moderate functionality, as this increases the proportion of hardware required by IFIS with respect to the function performed by the circuit.

Although the hardware overhead required by the IFIS methodology has been considerably reduced with the re-design of the individual IFIS cells, the overhead for the UART is still comparable to duplication. This is the best overhead possible using the duality solution, and arises because the function block within each IFIS cell is duplicated. If the ratio of the function block complexity to IFIS overhead is small, then the total system overhead will increase accordingly. Consequently, the IFIS methodology could be considered unsuitable for circuit designs which contain moderate or little functionality for this reason.

Chapter six introduced the internal re-design of the IFIS cells. From this it can be seen that each IFIS cell currently contains a single point of failure, that being the multiplexor selector output from the IFIS data control block. If the selector was faulty, the multiplexors would always offer the output data from the function block to the storage elements. The cell would thus continue to operate even if instructed to halt by other cells. Consequently, this represents a limitation to the current version of the IFIS methodology.

Finally, the IFIS methodology does not tolerate any disruption of the encoding scheme at all. This could be considered a disadvantage as several on-line test systems implement a 'rollback' strategy where calculations are re-tried several times in an attempt to recover from transient faults. The IFIS methodology cannot support this, and as such is more fault intolerant.

8.5 Measures of Success

The work contained within this thesis was successful in that the objectives stated in chapter three were reached with the experiments performed. The first objective was to investigate how the different encoding schemes and feedback systems impacted upon the data throughput and error halting ability of circuits designed using the IFIS methodology. This was successfully answered with the experiment described in chapter four as the data throughputs and error halting abilities were shown for several feedback structures. These feedback structures included self feedback, successor feedback, greater than successor feedback and a hybrid of self feedback and successor feedback. The results showed the impact of feedback structure on the halting ability and data throughput of designs employing the IFIS methodology.

The second objective was to identify which combination of feedback structures, encoding schemes and processing rules gave the greatest benefit in terms of data throughput, error halting ability and lowest implementation cost. The experiment described in chapter five implemented two encoding schemes and two processing rules with the best feedback structure identified in chapter four. The results allowed the identification of the most efficient combination of feedback structure, encoding scheme and processing rule, and thus also fulfilled the objective.

The third objective was to identify the factors which caused the significant hardware overhead within the IFIS cells, and re-design the internal circuitry such that this overhead was reduced. Four possible methods were identified which could be used as alternative architectures within the IFIS cell. The most generic solution of duality was selected for future use, which resulted in the hardware overhead dropping by almost a factor of two thereby fulfilling the objective.

The final objective was to evaluate the feasibility of the IFIS methodology using a commercial UART as a vehicle. The results obtained showed the IFIS methodology was feasible and could be used as an on-line test methodology provided the hardware overhead of 120% could be tolerated. As the UART was selected as being a realistic

and commercially available IC, the results can also be considered as realistic thereby fulfilling the objective.

The feasibility of the IFIS methodology has been demonstrated with the design of a commercial UART. The UART design has also addressed several issues including the design of complex state machines, combinational circuitry and the interfacing of IFIS designs with non-IFIS environments. Consequently, the IFIS UART represents one of the most complex on-line test designs investigated to date.

8.6 Limitations of the Work

The work presented within this thesis is limited in that only a small number of possible coding options were explored due to time restrictions upon the project. Also, the level at which the data encoding takes place could be seen to be a limitation as an information redundancy of 100% is immediately introduced. Consequently, the hardware implementation of this coding also exhibits similar properties with a overhead of 120% which is comparable to duplication with comparison.

The inability of the IFIS methodology to recover from transient faults where many other on-line test methodologies adopt a 're-try' approach could also be considered a limitation, as could the single point of failure within the IFIS cells due to the multiplexor selection line.

The IFIS cell re-design chapter detailed four possible solutions to the hardware overhead problem. These four solutions represent a limited set of possible solutions, but again due to time constraints the options had to be limited.

Finally, the IFIS UART could not be implemented to an ASIC cell library as the tools were not available. Consequently, ASIC characteristics for the IFIS UART could not be obtained and would indeed be of considerable interest. These points represent limitations of the current work and are worthy of further investigation.

8.7 Further Work

Further work that could be undertaken is a re-design of the IFIS data control block such that the output of the multiplexor is implemented as a two-rail design. This would eliminate the single point of failure from the IFIS cell and consequently lead to a more fault intolerant (i.e. better) design methodology.

Other work which could be undertaken is to improve the cell design such that the hardware overhead at the IFIS cell level is reduced even further. This should make the IFIS methodology more attractive for implementation in contemporary designs. Also, the implementation of the IFIS UART to an ASIC cell library could be undertaken to obtain ASIC characteristic values.

Finally, the methodology of IFIS lends itself very well to fault location as the parity and sequence in which the system outputs halt can uniquely identify the location of a fault to the IFIS cell boundary. This is also worthy of considerable attention, as the identification of an optimal algorithm for fault location within IFIS systems would be of great value should the IFIS methodology be adopted.

8.8 Summary of Thesis

This thesis has evaluated a novel parity based on-line test methodology designed to detect 'in the field' errors occurring from transient and permanent faults. The various encoding schemes, feedback structures and processing rules of this methodology have been evaluated, and the most efficient configuration chosen for implementation. This implementation has resulted in the design and test of a 'real life' UART design which has highlighted the feasibility and scalability of the IFIS methodology for digital systems design.

REFERENCES

- [Abadir, 1983] M. S. Abadir and H. K. Reghbati, "LSI Testing Techniques", IEEE Micro, February 1983, pp 34-51.
- [Aitken, 1991] R. C. Aitken, "Fault Location with Current Monitoring.", Proceedings of the IEEE International Test Conference, 1991, pp 623-632.
- [Asaad, 1996] H. A. Asaad, J. P. Hayes and B. T. Murray, "Design of Scaleable Hardware Test Generators for On-Line BIST.", 2nd IEEE International On-Line Test Conference, July 1996, pp 164-168.
- [Ashjaee, 1977] M. J. Ashjaee and S. M. Reddy, "On Totally Self-Checking Checkers for Separable Codes.", IEEE Transactions on Computers, Vol. C-26, No. 8, August 1977, pp 737-744.
- [Audet, 1996] D. Audet, N. Gagnon and Y. Savaria, "Quantitative Comparisons of TMR Implementations in a Multiprocessor System.", 2nd IEEE International On-Line Test Conference, July 1996, pp 196-199.
- [Avizienis, 1971] A. Avizienis, "Arithmetic Error Codes : Cost and Effectiveness Studies for Application in Digital Systems Design.", IEEE Transactions on Computers, Vol. C-20, No. 11, November 1971, pp 1322-1331.
- [Avra, 1993] L. J. Avra and E. J. McCluskey, "Synthesising For Scan Dependence in Built-In Self-Testable Designs.", International Test Conference, 1993, Paper 35.1.
- [Bahram, 1992] N. K. Bahram and D. Chakravarty, "ASIC Design Methodology Trends.", Electro / 92, Vol. 1, May 1992, pp 163-168.
- [Berger, 1961] J. M. Berger, "A Note on Error Detection Codes for Asymmetric Channels.", Information and Control, Vol. 4, 1961, pp 68-73.
- [Beucler, 1984] F. P. Beucler and M. J. Manner, "HILDO : The Highly Integrated Logic Device Observer.", VLSI Design, June 1984, pp 88-96.
- [Blaum, 1988] M. Blaum, "Systematic Unidirectional Burst Detecting Codes.", IEEE Transactions on Computers, Vol. C-37, No. 4, April 1988, pp 453-457.

- [Bleeker, 1993] H. Bleeker, "An Economic, Hands-On Start to Boundary Scan Testing.", Philips Industrial Electronics BV, Eindhoven, The Netherlands, Microprocessors and Microsystems, Vol. 17, No. 5, June 1993, pp 299-303.
- [Bolchini, 1996] C. Bolchini, F. Salice and D. Sciuto, "Design of Totally Self-Checking Checkers for a class of Hamming distance Codes.", 2nd IEEE International On-Line Test Conference, July 1996, pp 150-153.
- [Borden, 1982] J. M. Borden, "Optimal Asymmetric Error Detecting Codes.", Information and Control, Vol. 53, 1982, pp 66-73.
- [Bose, 1985] B. Bose and D. J. Lin, "Systematic Unidirectional Error-Detecting Codes.", IEEE Transactions on Computers, Vol. C-34, No. 11, November 1985, pp 1026- 1032.
- [Bose, 1986] B. Bose, "Burst Unidirectional Error-Detecting Codes.", IEEE Transactions on Computers, Vol. C-35, No. 4, April 1986, pp 350-353.
- [Brown, 1960] D. T. Brown, "Error Detecting and Correcting Binary Codes for Arithmetic Operations", IRE Transactions on Electronic Computers, September 1960, pp 333-337.
- [Burns, 1992] S. W. Burns and N. K. Jha, "A Totally Self-Checking Checker for a Parallel Unordered Coding Scheme.", IEEE VLSI Test Symposium, Paper 7.3, 1992.
- [Chang, 1996] W. F. Chang and C. W. Wu, "A TSC Berger-Code Checker for 2^{r-1} -Bit Information.", 2nd IEEE International On-Line Test Conference, July 1996, pp 158-161.
- [Cheema, 1992] M. S. Cheema and P. K. Lala, "Totally Self-Checking CMOS Circuit Design for Breaks and Stuck-On Faults.", Journal of Solid-State Circuits, Vol. 27, No. 8, August 1992, pp 1203-1206.
- [Corno, 1996] F. Corno, P. Prinetto and M. S. Reorda, "Coupling Genetic ATPG and Synthesis of Pattern Generations for Deterministic BIST.", 2nd IEEE International On-Line Test Conference, July 1996, pp 78-81.
- [David, 1995] I. David, R. Ginosar and M. Yoeli, "Self-Timed is Self-Checking.", Journal of Electronic Testing : Theory and Applications, 6, 1995, pp 219-228.

- [Dean, 1991] M. E. Dean, T. E. Williams and D. L. Dill, "Efficient Self-Timing with Level-Encoded 2-Phase Dual-Rail (LEDR).", *Advanced Research in VLSI*, 1991, pp 55-70.
- [Dean, 1994] M. E. Dean, D. L. Dill and M. Horowitz, "Self-Timed Logic Using Current-Sensing Completion Detection (CSCD).", *Journal of VLSI Signal Processing*, Vol. 7, 1994, pp 7-16.
- [Dong, 1984] H. Dong, "Modified Berger Codes for Detection of Unidirectional Errors.", *IEEE Transactions on Computers*, Vol. C-33, No. 6, June 1984, pp 572-575.
- [Ferguson, 1991] F. J. Ferguson and T. Larabee, "Test Pattern Generation for Realistic Bridge Faults in CMOS IC's.", *Proceedings of the IEEE International Test Conference*, 1991, pp 492-499.
- [Fertsch, 1991] M. T. Fertsch, S. H. Lee, J. Rioux, K. B. Sweetland, J. E. Watrous and P. N. Bompastore, "Design Considerations for CrossCheck Foundations and Libraries.", *Proceedings of the Fourth Annual IEEE International ASIC Conference and Exhibition*, P11-5/1-4.
- [Freiman, 1962] C. V. Freiman, "Optimal Error Detection Codes for Completely Asymmetric Binary Channels.", *Information and Control*, Vol. 5, 1962, pp 64-71.
- [Fujiwara, 1983] H. Fujiwara and T. Shimono, "On the Acceleration of Test Generation Algorithms.", *IEEE Transactions on Computers*, Vol. C-32, No. 12, December 1983, pp 1137-1144.
- [Gaitanis, 1996] N. Gaitanis, P. Kostarakis and A. Pashalis, "A Three Rail Totally Self-Checking Error Indicator.", *2nd IEEE International On-Line Test Conference*, July 1996, pp 50-52.
- [Garner, 1966] H. L. Garner, "Error Codes for Arithmetic Operations", *IEEE Transactions on Electronic Computers*, Vol. EC-15, No. 5, October 1966, pp 763-770.
- [Goel, 1981] P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits.", *IEEE Transactions on Computers*, Vol. C-30, No. 3, March 1981, pp 215-222.
- [Golstein, 1979] L. H. Goldstein, "Controllability / Observability Analysis of Digital Circuits.", *IEEE Transactions on Circuits and Systems*, Vol. CAS-26, No. 9, September 1979, pp 685-693.

- [Gössel, 1993] M. Gössel and E. S. Sogomonyan, "Code Disjoint Self-Parity Combinational Circuits for Self-Testing, Concurrent Fault Detection and Parity Scan Design.", *International Conference on Very Large Scale Integration*, Sept. 1993, Vol. A-42, pp 103-111.
- [Haider, 1993] N. S. Haider and N. Kanopoulos, "Efficient Board Interconnect Testing using the Split Boundary Scan Register.", *Journal of Electronic Testing : Theory and Applications*, Vol. 4, 1993, pp 181-189.
- [Hadlock, 1975] F. Hadlock, "Finding a Maximum Cut of a Planar Graph in Polynomial Time", *SIAM Journal on Computing*, Vol. 4, No. 3, September 1975, pp 221-225.
- [Hana, 1986] H. H. Hana and B. W. Johnson, "Concurrent Error Detection in VLSI Circuits using Time Redundancy", *Proceedings of the IEEE Southeastcon 1986 Regional Conference*, 1986, pp 208-212.
- [Hellerbrand, 1996] S. Hellebrand, H. J. Wunderlich and A. Hertwig, "Mixed Mode BIST using Embedded Processors.", *2nd IEEE International On-Line Test Conference*, July 1996, pp 82-85.
- [Jha, 1993] N. K. Jha and S.J. Wang, "Design and Synthesis of Self-Checking VLSI Circuits", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 12, No. 6, June 1993, pp 878-887.
- [Johnson, 1988] B. W. Johnson, J. H. Aylor and H. H. Hana, "Efficient Use of Time and Hardware Redundancy for Concurrent Error Detection in a 32-bit VLSI Adder.", *IEEE Journal of Solid-State Circuits*, Vol. 23, No. 1, February 1982, pp 208-214.
- [Jones, 1991] S. Jones and D. W. Lloyd, "Digital Test Methodology.", UK patent No. 9225327.8, 1991.
- [Khakbaz, 1984] J. Khakbaz and E. J. McCluskey, "Self-Testing Embedded Parity Checkers.", *IEEE Transactions on Computers*, Vol. C-33, No. 8, August 1984, pp 753-756.
- [Konemann, 1979] B. Konemann, J. Mucha and G. Zwierhoff, "Built-In Logic Block Observation Techniques.", *IEEE Test Conference*, 1979, Session 2, pp 37-41.
- [Kothari, 1993] R. D. Kothari and D.S. Ha, "Experimental Results on Aliasing Errors in Circular BIST Design.", *Proceedings of the ETC 93, Third European Test Conference*, IEEE Computer Society Press, April 1993, pp 466-474.

- [Larabee, 1989] T. Larabee, "Efficient Generation of Test Patterns Using Boolean Difference.", International Test Conference, 1989, Paper 35.1.
- [Li, 1992] J. Li and E. E. Swartzlander, "Concurrent Error Detection in ALU's by Recomputing with Rotated Operands", 1992 International Workshop on Defect and Fault Tolerance in VLSI Systems, 1992.
- [McCluskey, 1985] E. J. McCluskey, "Built-In Self-Test Techniques.", IEEE Design and Test, Vol. 2, No. 2, April 1985, pp 21-28.
- [McEuen, 1991] S. D. McEuen, "IDDq Benefits.", Proceedings of the IEEE VLSI Test Symposium, 1991, pp 285-290.
- [Metra, 1996] C. Metra and J. C. Lo, "Compact and High Speed Berger Code Checker.", 2nd IEEE International On-Line Test Conference, July 1996, pp 144-149.
- [Muehldorf, 1981] E. I. Muehldorf and A. D. Savkar, "LSI Logic Testing - An Overview", IEEE Transactions on Computers, Vol. 30, No. 1, January 1981, pp 1-16.
- [Nigh, 1990] P. Nigh and W. Maly, "Test Generation for Current Testing.", IEEE Design and Test of Computers, February 1990, pp 26-38.
- [Nikolos, 1996] D. Nikolos, "Optimal Self-Testing Embedded Two-Rail Checkers.", 2nd IEEE International On-Line Test Conference, July 1996, pp 154-157.
- [Patel, 1982] J. H. Patel and L. Y. Fung, "Concurrent Error Detection in ALU's by Recomputing with Shifted Operands.", IEEE Transactions on Computers, Vol. C-31, No. 7, July 1982, pp 589-595.
- [Piestrak, 1996] S. J. Piestrak, "Modular Design of Self-Testing Checkers for m-out-of-n Codes.", 2nd IEEE International On-Line Test Conference, July 1996, pp 132-136.
- [Reddy, 1985] M. K. Reddy and S. M. Reddy, "Transistor Level Test Generation for MOS Circuits.", Proceedings of the 22nd Design Automation Conference, 1985, pp 825-828.
- [Reynolds, 1978] D. A. Reynolds and G. Metze, "Fault Detection Capabilities of Alternating Logic.", IEEE Transactions on Computers, Vol. C-27, No. 12, December 1978, pp 1093-1098.

- [Roth, 1966] J. P. Roth, "Diagnosis of Automata Failures : A Calculus and a Method.", IBM Journal of Research and Development, Vol. 10, July 1966, pp 278-291.
- [Roth, 1969] J. P. Roth, W. G. Bouricius and P. R. Schneider, "Programmed Algorithms to Compute Test to Detect and Distinguish Between Failures in Logic Circuits.", IEEE Transactions on Electronic Computers, Vol. EC-16, No. 5, October 1969, pp 567-580.
- [Sahni, 1976] S. Sahni and T. Gonzalez, "P-Complete Approximation Problems.", Journal of the Association for Computing Machinery, Vol. 23, No. 3, July 1976, pp 555-565.
- [Seth, 1985] S. C. Seth and V. D. Agrawal, "Cutting Chip Testing Costs.", IEEE Spectrum, April 1985, pp 38-45.
- [Shedletsky, 1978] J. J. Shedletsky, "Error Correction by Alternate-Data Retry.", IEEE Transactions on Computers, Vol. C-27, No. 2, February 1978, pp 106-112.
- [Shultz, 1989] M. H. Shultz and E. Auth, "Improved Deterministic Test Pattern Generation with Applications to Redundancy Identification.", IEEE Transactions on Computer-Aided Design, Vol. 8, No. 7, July 1989, pp 811-816.
- [Smith, 1984] J. E. Smith, "On Separable Unordered Codes", IEEE Transactions on Computers, Vol. C-33, No. 8, August 1984, pp 741-743.
- [Sutherland, 1989] I. Sutherland, "Micropipelines.", Communications of the ACM, Vol. 32, No. 6, June 1989, pp 720-733.
- [Wakerly, 1973] J. F. Wakerly, "Partially Self Checking Circuits and their use in Performing Logical Operations.", Proceedings of the 3rd Fault Tolerant Computing Symposium, pp 65-70.
- [Williams, 1983] T. W. Williams and K. P. Parker, "Design for Testability - A Survey", Proceedings of the IEEE, Vol. 71, No. 1, January 1983, pp 98-112.
- [Zorian, 1993] Y. Zorian and A. Ivanov, "Programmable Space Compaction for BIST.", Digest of Papers, FTCS-23, The twenty third International Symposium on Fault Tolerant Computing, IEEE Computer Society Press, August 1993, pp 340-349.

BIBLIOGRAPHY

- [Antoniou, 1993] A. Antoniou, "Digital Filters - Analysis, Design and Applications.", McGraw-Hill International (Second Edition), 1993, ISBN 0-07-002121-X.
- [Bennetts, 1984] R. G. Bennetts, "Design of Testable Logic Circuits.", Addison-Wesley, 1984, ISBN 0-201-14403-4.
- [Gossel, 1993] M. Gossel and S. Graf, "Error Detection Circuits", McGraw-Hill International (UK) Limited, 1993, ISBN 0-07-707438-6
- [Lala, 1985] P. K. Lala, "Fault Tolerant and Fault Testable Hardware Design.", Prentice Hall International, 1985, ISBN 0-13-308248-2.
- [Mead, 1980] C. Mead and L. Conway, "Introduction to VLSI Systems.", Addison-Wesley Publishing Company, 1980, ISBN 0-201-04358-0
- [Piestrak, 1995] S. J. Piestrak, "Design of Self-Testing Checkers for Unidirectional Error Detecting Codes", Institute of Cybernetics Engineering, Technical University of Wroclaw, 1995, ISSN 0324-9786.
- [Rajsuman, 1992] R. Rajsuman, "Digital Hardware Testing : Transistor Level Fault Modelling and Testing.", Artech House, 1992, ISBN 0-89006-580-2.
- [Russell, 1989] G. Russell and L. Sayers, "Advanced Simulation and Test Methodologies for VLSI Design", Van Nostrand Reinhold (International), 1989, ISBN 0-7476-0001-5.
- [Siewiorek, 1992] D. P. Siewiorek and R. S. Swarz, "Reliable Computer Systems", Second Edition, Digital Press, 1992, ISBN 1-55558-075-0.

PUBLISHED PAPERS

- [Saeed, 1996] M.Saeed, D. Thulborn, J. Yeandel and S. Jones, "IFIS - An On-Line Testing Methodology Using Dual-Rail Data Coding.", 2nd IEEE International On-Line Test Conference, Bairritz, France, July 1996, pp 68-72.
- [Yeandel, 1996] J. Yeandel, D. Thulborn, M.Saeed and S. Jones, "Fault Localisation for On-Line Testable Designs Realised Using Dual-Rail Design Methodology.", 2nd IEEE International On-Line Test Conference, Bairritz, France, July 1996, pp 221-222.
- [Yeandel, 1996] J. Yeandel, D. Thulborn and S. Jones, "An On-Line Testable UART Implemented Using IFIS.", 15th IEEE VLSI Test Symposium, Monterey, California, April 1997.

