The Dissertation Committee for Avijit Dutta Certifies that this is the approved version of the following dissertation:

Synthesis for Circuit Reliability

Committee:

Nur Touba, Supervisor

Tony Ambler

Margarida Jacome

Zhigang Pan

Abhijit Jas

Synthesis for Circuit Reliability

by

Avijit Dutta, B.E.; M.S.E.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

May, 2007

Dedication

My thanks go, as always, to my maternal grandparents, to whom this dissertation is dedicated. I am indebted to my maternal grandparents for standing beside me all these years and constantly motivating me in my efforts. I wouldn't have been what I am, without their help.

And

My thanks go to my supervising professor, Nur A. Touba.

Acknowledgements

I wish to thank my advisor, Professor Nur A. Touba for his continued guidance and encouragement. His research philosophy of maintaining a balance between pure theoretical and practical aspects of research has influenced me a lot.

Finally, my thanks go to the other committee members, Prof. Margarida Jacome, Prof. Tony Ambler, Prof. David Z. Pan and Dr. Abhijit Jas for their input and support.

Synthesis for Circuit Reliability

Publication No._____

Avijit Dutta, Ph.D.

The University of Texas at Austin, 2007

Supervisor: Nur A. Touba

For modern logic circuits, circuit reliability is an important design consideration. Ionizing radiation from high-energy neutrons and alpha particles can cause a *single-event upset (SEU)* that may cause a bit flip in some latch or memory element thereby altering the state of the system resulting in a *soft error*. As process technology scales well below 100 nanometers, the higher operating frequencies, lower voltage levels, and smaller noise margins make integrated circuits increasingly susceptible to SEUs resulting in a dramatic increase in soft errors. In this dissertation, a non-intrusive technique is presented to detect soft errors in multilevel combinational logic circuit with minimal overhead. Another low cost error correcting code based technique is presented to detect and correct the most likely soft errors in memory. This technique is then extended to design a low cost unequal error protection code which can protect data residing in a router buffer effectively. The dissertation also contains a fast algorithm to accurately estimate signal probabilities of circuit lines. This algorithm can be used to estimate soft error rates in a logic circuit. Finally, the

dissertation also includes a low cost test data compression technique to reduce the deterministic test data to be stored on tester during off-line testing of a circuit.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Circuit reliability has become an important design consideration. When ionizing radiation from high-energy neutrons and alpha particles strike a sensitive region in a semiconductor device, they generate a dense track of electron-hole pairs that may be collected by a p-n junction resulting in a very short duration pulse of current causing a *single-event upset (SEU)* in the signal value. An SEU may cause a bit flip in some latch or memory element thereby altering the state of the system resulting in a *soft error*. As process technology scales well below 100 nanometers, the higher operating frequencies, lower voltage levels, and smaller noise margins make integrated circuits increasingly susceptible to SEUs resulting in a dramatic increase in soft errors. Studies indicate that the soft error failure rate will become unacceptable even in mainstream commercial applications. It is projected that soft errors in logic circuits will be the limiting factor for system reliability as technology continues to scale. The problem of soft error is even more prominent in memories. Constant technology process improvement has resulted in very dense memory cells that store information with less capacitance and lower voltage. Consequently, less charge is required to produce one or more soft errors in memories. While off-line tests can detect manufacturing defects, on-line error detecting and correcting schemes are required to detect soft-errors and recover from soft-errors. Based on the current technological trends, there is a great need for concurrent error detection and correction techniques to increase reliability of both combinational logic circuits and memories.

This dissertation primarily addresses the threat to reliability arising from increasing soft error rates and proposes several concurrent error detection and correction methodologies. For off-line testing, the dissertation presents a very low cost solution to

effectively reduce the deterministic test data to be stored for manufacturing test of a circuit.

This chapter provides background on the issues related to soft errors in logic circuit as well as in memories. It also presents the problem of test data compression and its necessity in the context of off-line manufacturing test. Section 1.1 describes the reliability concerns in combinational logic due to soft errors and describes the need for efficient concurrent error detection techniques. Section 1.2 describes the problems in memories caused by single event induced multiple bit upsets and present a low cost error correcting code to design multiple bit upset tolerant memories. Section 1.2 also describes a low cost unequal protection code that can protect data residing in router memories by providing more protection to the more important part of the data packet. Section 1.3 presents a memory efficient algorithm with low runtime complexity to accurately estimate signal probability bounds of the circuit lines. This algorithm can be used for accurate estimation of soft error rates in combinational logic circuits. Accurate estimate of soft error rate can help in insertion of appropriate protection hardware in logic circuits. In section 1.4 the problem of test data compression in offline testing is described. In this section a limited depth sequential expansion based test data compression technique is presented to reduce both test data and testing time.

1.1 CONCURRENT ERROR DETECTION

One way to detect soft errors is to use concurrent error detection (CED) circuitry that monitors the outputs of a circuit for the occurrence of an error. If an error is detected, then the system can recover thereby preventing a failure. Detecting errors in logic circuits is much more expensive than in memories. While CED can be efficiently incorporated in memories due to their regular structure, logic circuits with their irregular structure present a much greater challenge. It is projected that in systems where

memory CED is employed, soft errors in logic circuits will be the limiting factor for system reliability as technology continues to scale. In this dissertation we focus on the problem of providing CED in logic circuits. The simplest CED scheme for logic circuits is to use duplication where the circuit is duplicated and the outputs are compared with an equality checker. While this is very simple to implement and provides very high error coverage, it requires over 100% overhead. So there is a need for CED schemes that provide high coverage of soft-errors and at the same time have low overhead. A new method for synthesizing non-intrusive concurrent error detection (CED) circuitry is presented. The idea is to use single-bit parity to detect all errors affecting an odd number of bits and then synthesize a circuit to detect the even errors. A novel statistical sampling and expanding methodology is proposed for constructing the even error detection circuitry. A major feature of the proposed methodology is that it allows very efficient tradeoffs between error coverage and overhead. While CED schemes that use a fixed checker based on a particular error detecting code are not amenable to simplification without a major impact on coverage, the proposed scheme can easily facilitate significant reductions in overhead with only a small loss in coverage. Experimental results show that the proposed scheme can provide very high levels of soft error protection at a fraction of the cost of duplication. In chapter 2, detailed description of the proposed non-intrusive concurrent error detection scheme for combinational logic circuit is provided.

1.2 LOW COST ERROR CORRECTING CODES FOR MEMORIES

Conventional error correcting code (ECC) schemes used in memories and caches cannot correct double bit errors caused by a single event upset (SEU). As memory density increases, multiple bit upsets in nearby cells become more frequent. In this dissertation, we propose a methodology for deriving an error correcting code through heuristic search that can detect and correct the most likely double bit errors in a memory

while minimizing the miscorrection probability of the unlikely double bit errors. A key feature of the proposed ECC is that it uses the same number of check bits as the conventional single error correcting/double error detecting (SEC-DED) codes commonly used, and has nearly identical syndrome generator/encoder area and timing overhead. Hence, there is very little additional cost to using the proposed ECC. The proposed ECC scheme can be very useful for small memories e.g., content addressable memory (CAM), register files where interleaving is not possible. The proposed ECC can be used instead of or in addition to bit interleaving to provide greater flexibility for optimizing a memory layout and/or provide better protection from multiple bit upsets. The proposed code designs the parity check matrix of the linear block code while avoiding certain types of linear dependencies involving the columns of the parity check matrix. This selective linear dependency avoidance based algorithm is further extended to design a low cost unequal error protection code. This kind of code can be used to protect router memories from single event induced multiple bit upsets. The network-on-chip (NoC) paradigm is seen as a way of facilitating the integration of a large number of computational and storage blocks on a chip to meet several performance and power constraints. However due to continued scaling of process technologies, the devices and interconnects have become more sensitive to new types of reliability hazards such as, single event upsets and crosstalk. This dissertation presents a low cost error correcting code based techniques to protect the NoC routers against the single event upset induced soft errors and also against crosstalk. An unequal protection error correcting code based methodology is provided for the most commonly used store-and-forward routing strategy. The proposed code has the same check bit overhead as the conventional single error correcting (SEC) code. The encoding/decoding overhead and latency are also similar to the conventional low cost SEC code. The proposed codes belong to the class of unequal error protection codes as

they provide different levels of error correction capability for different portions of the same packet. The proposed code provides more protection for the important parts of the data. Chapter 3 and 4 describes the proposed codes in detail and also provides detailed comparison with the existing codes.

1.3 ACCURATE SIGNAL PROBABILITY ESTIMATION

Estimating the reliability of circuits is essential in synthesis of reliable circuits. Accurate estimation of circuit reliability requires accurate estimation of soft-error rates. The effects of single event upsets (SEU) on digital circuit can be categorized in three ways: 1) SEUs can cause a transient error in combinational logic part which can be propagated and captured in flip-flops. 2) SEUs can directly change the contents of memory elements. 3) SEUs can cause permanent damage on SRAM based combinational circuits e.g., FPGAs. In our research so far, we have looked into the first category involving combinational circuits and their effect on circuit reliability. An SEU causes a soft-error if and only if it propagates to the latch boundary and gets latched making a bit-flip error. Most SEUs on combinational logic are masked and they don't reach the latch boundary. Moreover even if the SEU effect reaches the latch boundary it still may not be captured if it does not reach the latch at the appropriate latching-window. Accurate estimation of signal probability is required to find out whether the effect of an SEU will propagate to the flip-flop inputs through the combinational gates. The effect can only propagate if for all the gates on the propagation path all the other inputs have non-controlling logic values. To calculate actual soft-error probability accurate estimation of signal probabilities on the propagation path of the SEU effect is required. In this proposal, we present an efficient method to accurately compute tight bounds on the signal probabilities for combinational circuits. This dissertation presents an improved method to accurately estimate signal probabilities using ordered partial decision diagrams (OPDDs)

[Kodavarti 93] for partial representation of the functions at the circuit lines. OPDDs which are limited to a certain maximum number of nodes are built iteratively with different variable orderings to efficiently explore different regions of the function. Signal probability bounds (upper and lower) are computed from the OPDDs. From each OPDD, information is extracted to tighten the signal probability bound and guide the variable ordering for the next OPDD. By restricting the size of each OPDD to a small number of nodes, they can be constructed and processed quickly to obtain a fast and accurate estimate of signal probabilities. Experimental results demonstrate the effectiveness of the approach compared with existing methods. Chapter 5 describes in detail the proposed approach.

1.4 TEST DATA COMPRESSION

In this dissertation till now we have focused on online methodologies to design reliable circuits. The proposed concurrent error detection/correction methodologies primarily focus on protecting circuits from single event induced soft errors. Another threat to circuit reliability is the manufacturing defects. Offline testing typically screens off the defective circuits. Offline testing techniques target the permanent faults and deterministic test data targeting the permanent faults are typically stored on the tester. For offline testing the test data volume and test time are the two major concerns. In this dissertation we present a low overhead input data compression technique to reduce the deterministic test data that needs to be stored on the tester. The proposed scheme can reduce the test time significantly. The proposed technique uses a limited depth sequential expansion based approach and incorporates the constraints directly into ATPG backtrace.

Existing techniques that incorporate decompressor constraints in the ATPG search/backtrace (e.g., Illinois scan) are based on combinational expansion in which each scan slice must be encoded using only the free-variables arriving form the tester in the

current clock cycle. Sequential expansion is more powerful as it allows free-variables across multiple clock cycles to be used, however conventional approaches for sequential expansion that are based on linear finite state machines (LFSRs) and ring generators are not amenable to including the constraints in the ATPG backtrace because the constraints are too complex. This dissertation investigates the use of limited dependence sequential expansion to combine the benefits of sequential decompression with the benefits of incorporating the decompressor constraints in the ATPG backtrace. Analytical and experimental results are presented showing the benefits of the proposed approach. Chapter 5 describes the proposed compression technique in detail.

Chapter 2: Synthesis of Non-Intrusive Concurrent Error Detection Using an Even Error Detecting Function

A new method for synthesizing non-intrusive concurrent error detection (CED) circuitry is presented. The idea is to use single-bit parity to detect all errors affecting an odd number of bits and then synthesize a circuit to detect the even errors. A novel statistical sampling and expanding methodology is proposed for constructing the even error detection circuitry. A major feature of the proposed methodology is that it allows very efficient tradeoffs between error coverage and overhead. While CED schemes that use a fixed checker based on a particular error detecting code are not amenable to simplification without a major impact on coverage, the proposed scheme can easily facilitate significant reductions in overhead with only a small loss in coverage. Experimental results show that the proposed scheme can provide very high levels of soft error protection at a fraction of the cost of duplication.

2.1 RELATED WORK

When ionizing radiation from high-energy neutrons and alpha particles strike a sensitive region in a semiconductor device, they generate a dense track of electron-hole pairs that may be collected by a p-n junction resulting in a very short duration pulse of current causing a *single-event upset (SEU)* in the signal value. An SEU may cause a bit flip in some latch or memory element thereby altering the state of the system resulting in a *soft error*. Additionally, an SEU may occur in an internal node of combinational logic and subsequently propagate to and be captured in a latch. As process technology scales well below 100 nanometers, the higher operating frequencies, lower voltage levels, and smaller noise margins make integrated circuits increasingly susceptible to SEUs resulting

in a dramatic increase in soft errors. Studies indicate that the soft error failure rate will become unacceptable even in mainstream commercial applications [Ziegler 96], [Cohen 99]. One way to detect soft errors is to use concurrent error detection (CED) circuitry that monitors the outputs of a circuit for the occurrence of an error [Gössel 93], [Nicolaidis 98]. If an error is detected, then the system can recover thereby preventing a failure. Detecting errors in logic circuits is much more expensive than in memories. While CED can be efficiently incorporated in memories due to their regular structure, logic circuits with their irregular structure present a much greater challenge. It is projected that in systems where memory CED is employed, soft errors in logic circuits will be the limiting factor for system reliability as technology continues to scale [Shivakumar 02], [Bowman 03, 04]. This chapter focuses on the problem of providing CED in logic circuits. The simplest CED scheme for logic circuits is to use duplication where the circuit is duplicated and the outputs are compared with an equality checker. While this is very simple to implement and provides very high error coverage, it requires over 100% overhead. A lot of research has been done on alternate schemes that are still applicable to any logic circuit but require less hardware overhead than duplication. One class of techniques uses time redundancy. Multiple sampling of the outputs has been proposed in [Franco 94], [Metra 98], [Nicolaidis 99], [Favalli 02]. Self-dual functions have been proposed in [Saposhnikov 96, 98a]. These approaches have low hardware costs, but reduce performance. Another class of techniques involves re-synthesizing the functional logic so that it has a more regular structure such that simple error detecting codes can be used to provide high coverage. Techniques have been developed for parity codes [De 94], [Touba 97], [Bolchini 97]; Berger codes [Jha 93], [Saposhnikov 98b]; and Bose-Lin codes [Das 99]. In cases where it is not desirable to re-synthesize the functional logic (e.g., cores, macrocells, handcrafted designs, legacy designs, etc.), these

techniques are not applicable. A third class of techniques uses non-intrusive CED where the functional logic is not modified. As shown in [Gössel 93], this problem can be formulated as follows (see illustration in Fig. 2.1). For a functional circuit with n inputs, $A=a_1,\ldots,a_n$, and m outputs $Z=z_1,\ldots,z_m$, let $EDF(a_1,\ldots,a_n, z_1,\ldots,z_m)$ be the error detecting function which is a Boolean function that is equal to 0 if the output vector Z is error-free, equal to 1 if the output vector Z has an error due to a fault in the specified fault class, and equal to X (don't care) in all other cases (i.e., for input vector A, no fault can cause the output vector to be equal to Z). Any implementation of the Boolean function *EDF* will detect all errors due to the specified fault class. As pointed out in [Almukhaizim 04a], the *EDF* could be passed directly to a synthesis tool to produce the CED circuitry and if the synthesis algorithm could search exhaustively, it could find the optimal non-intrusive CED circuit. However, synthesis tools use heuristics to search the large space of solutions and consequently may obtain a sub-optimal solution. In fact the nature of the *EDF* function makes it particularly hard for synthesis tools to handle as it has a very large don't care space and many exclusive-or (XOR) factors which most synthesis tools are not good at finding. Thus, passing the *EDF* directly to a synthesis tool generally does not produce good results as shown in [Almukhaizim 04a]. Rather than trying to directly synthesize the *EDF*, researchers have explored structured implementations for the *EDF*. The basic approach for this is to place a compaction circuit at the outputs of the function logic to reduce them from m down to k and then synthesize a prediction circuit that independently predicts the k outputs. This is illustrated in Figure 2.2. One approach for compacting the outputs is to use a parity code which XORs together different subsets of the outputs [Sogomonyan 74], [Fujiwara 87]. If the parity code is selected so that no errors are masked, then 100% coverage can be maintained. In [Almukhaizim 04b], it was observed that the overhead for using a parity code is

dominated by the prediction logic and a method based on entropy was proposed to guide the selection of the parity code to minimize the prediction logic. A technique for selecting the parity code with bounded error masking was described in [Tarnick 94]. In [Almukhaizim 04a], a more general design methodology that is not limited to parity was described for synthesizing the compaction circuit to ensure no error masking. In [Mohanram 03], CED based on a parity code is selectively disabled for some input vectors to tradeoff less coverage for less overhead in the prediction logic. In [Morozov 00], a technique for using a Berger code was described.

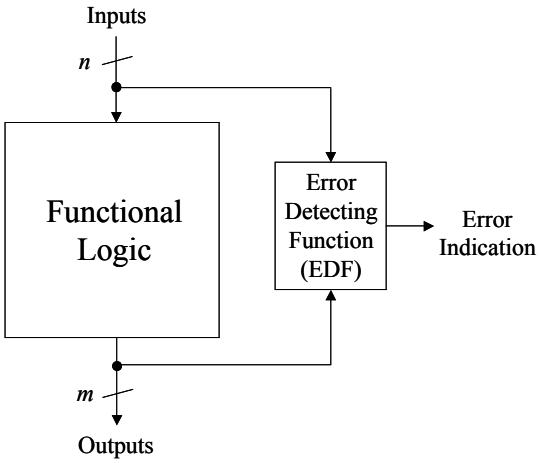

Figure 2.1: Non-intrusive CED

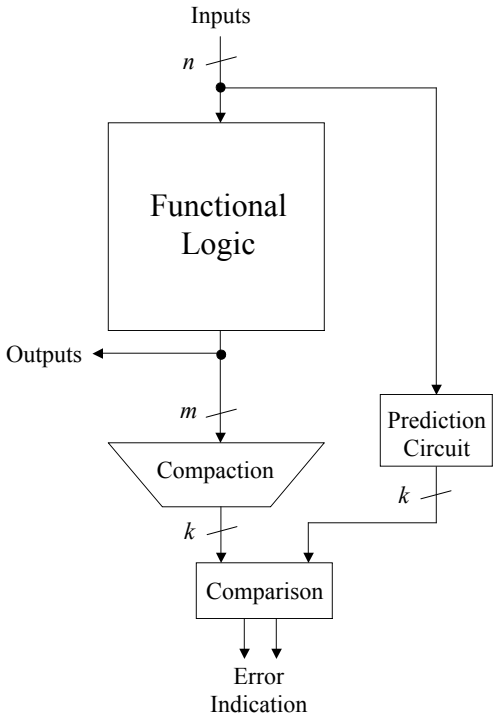

Figure 2.2: Basic approach for structured implementation of non-intrusive CED

In this chapter, a new method for synthesizing non-intrusive CED circuitry is presented. The idea is to use single-bit parity to detect all errors affecting an odd number of bits and then synthesize a circuit to detect the even errors. The key concept behind this approach is that most of the errors in the *EDF* function are single-bit errors. By using single-bit parity, all of the odd errors in the *EDF* function (which includes the single-bit errors) become don't cares leaving only the even errors. The smaller number of even errors in the *EDF* function can be efficiently synthesized with most synthesis tools. In effect, the proposed method forces a decomposition of the *EDF* function in which the odd errors are covered with a single parity function and the even errors are covered via conventional logic synthesis with don't cares. Forming the *EDF* function for even errors by exhaustive simulation of all input vectors and all faults can be done only for small circuits. In order to handle larger circuits, a novel statistical sampling and expanding methodology is proposed. While most CED schemes use a fixed checker structure based on an error detecting code that it not amenable to simplification without a significant impact on error coverage. One of the nice features of the proposed scheme is that it provides very easy and efficient tradeoffs between coverage and overhead. A systematic approach is described for simplifying the even error detecting function that results in large reductions in overhead with only a minor loss in error coverage. The chapter is organized as follows: Sec. 2.2 provides an overview of the proposed scheme and its architecture. Sec. 2.3 describes the procedure for forming the even error detecting function. Sec. 2.4 explains how the proposed scheme allows for very efficient tradeoffs in coverage versus overhead. Experimental results are presented in Sec. 2.5. Section 2.6 concludes the chapter.

2.2 OVERVIEW OF PROPOSED SCHEME

The proposed scheme involves combining single-bit parity with an even error detecting circuit. A block diagram for the proposed approach is shown in Fig. 2.3. The even error detecting circuit generates a two-bit error indication signal which normally has opposite values in the fault-free case and indicates an error by having equal values. An XOR-tree is used to compute the parity of the outputs of the functional logic. The parity predictor circuit predicts the complement of the parity of the outputs such that its output together with the XOR-tree output forms a two-bit error indication. The two pairs of error indication signals are then merged using a two-rail checker.

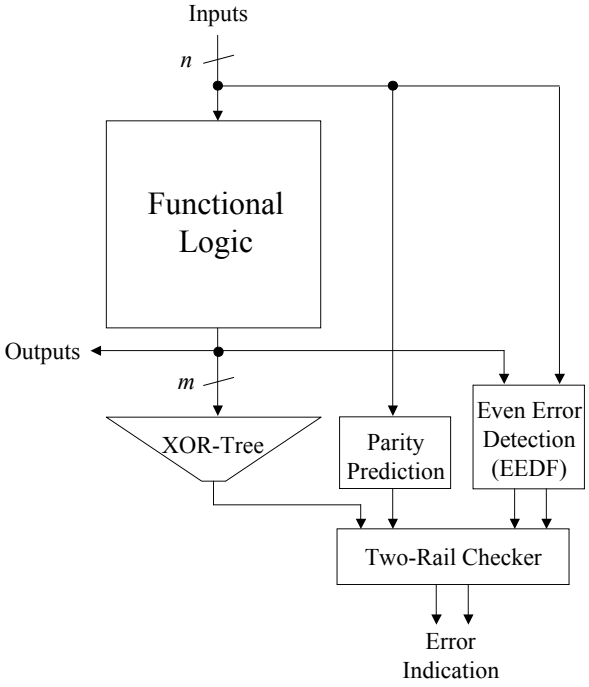

Figure 2.3: Block diagram of proposed scheme

To simplify things, the even error detection function (*EVEN_EDF*) will be described in the rest of the chapter as a single output function. The process of converting it so that it

produces a two-bit error indication signal is trivial. It can be done by simply extracting one XOR factor, inverting it, and making it a separate output (i.e., extract any factor *E2* such that *EVEN_EDF=E1⊕ E2* and use *E1* and *E2'* as outputs with the XOR gate removed). Thus, anytime *EVEN_EDF* is a 1, *E1* and *E2'* will have equal values indicating an error, and anytime *EVEN_EDF* is a 0, *E1* and *E2'* will have opposite values which is the normal error-free case. Synthesizing the parity predictor circuit is exactly the same as for previously proposed methods. Synthesizing the even error detecting circuit is done by forming the *EVEN_EDF* function and giving it to a synthesis tool to synthesize. The challenge is how to form the *EVEN_EDF* function and that is the subject of the next section.

2.3 FORMING EVEN_EDF FUNCTION

Given a functional logic circuit *F* with *n* inputs and *m* outputs, the simplest way to get the complete *EVEN_EDF* function that provides 100% coverage of all errors would be to exhaustively simulate *F* for all input vectors and faults. For each input vector, each fault is injected and the corresponding faulty output vector is obtained. If the faulty output vector has an even number of errors, then the minterm corresponding to the input vector and faulty output vector pair would be added to the ON-set of the *EVEN_EDF* function. This would continue until the complete ON-set for *EVEN_EDF* is formed. The OFF-set for *EVEN_EDF* is described by the functional logic circuit *F* itself. The DC-set includes anything that is not in the ON-set or OFF-set.

Forming the exact *EVEN_EDF* function through exhaustive simulation is intractable for all but the smallest circuits. Thus a less computationally complex procedure needs to be used for forming the *EVEN_EDF* function which will not necessarily obtain the exact ON-set. The proposed method involves using statistical methods to approximate the ON-set. Fortunately, good results can still be obtained even

if the exact ON-set is not known. If some extra minterms from the DC-set are included in the ON-set, there is no loss of coverage, but possibly the synthesis tool may not obtain as optimal of a result. If some minterms are missing from the ON-set, there may be some loss of coverage. If the approximate ON-set is reasonably close to the exact ON-set, the impact in terms of either the optimality of the synthesis or the coverage can be kept very small. Moreover, if one is interested in trading off less coverage for less overhead, this can be nicely facilitated by approximating the ON-set in a way that the missing minterms simplify the logic implementation.

The proposed method for approximating the ON-set of the *EVEN_EDF* function involves random sampling of the input space for each fault combined with a *bit-stripping* operation. The procedure is described as follows:

Input: Functional logic circuit *F*, fault list, and number of simulations to do per fault (*L*).

Output: Approximate ON-set for *EVEN_EDF* function.

Step 1: Prune fault list – All faults that have a structural path to only one output are pruned from the fault list as they will never cause even errors.

Step 2: Randomly simulate *L* input vectors for each fault in fault list – The value of *L* is a parameter for this procedure that allows tradeoffs between runtime versus accuracy.

Step 3: For any vector that causes an even error, perform bit-stripping – Select a bit in the input vector and flip its value to the opposite of its current value and fault simulate. If the error is no longer even, then the input bit is flipped back to its original value. Otherwise, the input bit is changed to an X since an even error occurs regardless of the value of that input bit. This process is repeated for all the bits in the input vector one by one. The order in which the bits are processed is selected randomly each time a new

vector is processed so that no particular order is repeated. The purpose of bit-stripping is to convert the input vector into an input cube that covers a large set of minterms.

Step 4: Add to the ON-set each input cube obtained in step 3 along with its corresponding output cube – Each input cube found in step 3 is fault simulated to obtain its corresponding 3-valued output cube. Together they specify a cube of minterms that are added to the ON-set of the *EVEN_EDF* function.

The procedure above produces an approximation of the ON-set for the *EVEN_EDF* function. Rather than simulating all of the input vectors for each fault (which would be exponential), only L vectors are simulated per fault where L is a user-specified value based on the desired level of accuracy in approximating the ON-set. Each input vector that causes an even error is expanded into a cube using bit-stripping. The resulting cube after bit-stripping contains many input vectors that also cause an even error. Some input vectors that cause an even error may not be found using this procedure because they may not be contained in any of the input cubes generated through bit-stripping. The larger the value of L, the more input cubes that are generated per fault and hence the less chance of missing an input vector that causes an even error for the fault. Missing input vectors that cause even errors means the ON-set for the *EVEN_EDF* function will be missing minterms which may result in some loss of coverage. However, on the good side, the minterms that are included in the *EVEN_EDF* function are contained in cubes (due to the way they were generated) and thus may simplify the logic implementation of the approximate *EVEN_EDF* function compared to the exact *EVEN_EDF* function that gives 100% coverage. The other source of approximation in the procedure is that one output cube is associated with all the input vectors contained in an input cube. In reality, of course, each input vector corresponds to only a single output vector and not a whole cube of output vectors. While the output cube is guaranteed to contain the correct output

vector, it also contains many other output vectors thereby resulting in extra minterms being placed in the ON-set which should actually be in the DC-set. There is no risk of any minterms from the OFF-set ending up in the ON-set since the output cube always contains an even error (this is ensured by the way the bit-stripping is done) and thus it can never contain a fault-free output vector. The fact that the approximate ON-set contains some minterms from the DC-set does not impact the coverage at all. Potentially it could make the logic implementation of the approximate EVEN_EDF function more complex compared with the exact EVEN_EDF, but the fact that the additional minterms in the ON-set are contained in cubes (due to the way they were generated) the impact generally will not be significant.

Even though the experiments are performed with single stuck-at fault model, the proposed algorithm behaves in a conservative way for transient faults. If a transient fault is such that it causes an odd number of errors at the output when the corresponding stuck-at fault causes an even number of errors, the fault will still be detected by the odd error detection circuit. This scenario may arise when the transient fault propagates to the outputs only through some of the possible paths (shorter paths) due to the transient nature of the fault. On the other hand if an even number of errors are caused at the output when the corresponding stuck-at fault causes an odd number of errors, by the transient fault, then there could be some loss of coverage because the corresponding input vector and faulty output pair was not included in the ON-set of the EVEN_EDF function. However not all even errors are possible for each odd error. It will depend on the distribution of the long paths and short paths and probability of transient errors. Therefore there can only be minimal loss in coverage. Extensive path length analysis and transient error probabilities will be required to analyze the actual probability of transient errors causing an odd error to degenerate into even error. The faults in the CED circuitry will also be detected

because the CED circuitry is an irredundant part of the final synthesized circuit and we are considering only single fault at a time. Some patterns will eventually uncover the fault in the CED part. False alarms i.e. flagging error when there is no error are also not possible since the bit-stripping procedure ensures that none of the fault-free response vector and input vector combinations is part of the ON-set of the EVEN_EDF function. Since the procedure is based on pure random sampling, no special ATPG is required to construct the EVEN_EDF function. The proposed algorithm provides a very easy and efficient trade-off between simulation time and fault coverage by controlling the parameter *L*, while constructing the EVEN_EDF function.

2.4 COVERAGE VERSUS AREA OVERHEAD TRADEOFFS

If the coverage is not high enough, the procedure described in Sec. 2.3 can be repeated with a larger value of L to obtain a more accurate approximation of the EVEN_EDF function and then the even error detecting circuit can re-synthesized.

If lower overhead is desired for the CED circuitry, a strategy for achieving this while minimizing the loss of coverage is as follows. When the input cubes are generated via bit-stripping in Step 3 of the procedure described in Sec. 2.3, a threshold can be set on the size of the cubes. If the size of the input cube is not larger than the threshold, then the input cube is simply discarded and not added to the ON-set. The reasoning behind this strategy is that small input cubes contain only a small number of input vectors while requiring a potentially large amount of logic to implement (depending on the extent to which they can be merged or factored with other cubes). By discarding these cubes, the impact on the overall coverage is minimal while the benefit in reducing overhead is substantial. This strategy can be very effective in trading off a small loss in coverage for a large reduction in overhead. This is one of the key advantages of the proposed schemes and will be highlighted in the experimental results.

2.5 EXPERIMENTAL RESULTS

Experiments were performed on some MCNC benchmark circuits [Yang 91]. The area results for the circuits were obtained using Synopsis Design Analyzer. The area reported is the cell area.

Table 2.1 compares the area overhead for the self-checking circuits implemented using the duplication method and the proposed scheme. Both are non-intrusive and hence do not require re-synthesis of the functional logic. The circuit information and the optimized area for the MCNC benchmark circuits with no CED can be found under the first major heading. Under the second and third major headings the results corresponding to the duplication method and the proposed scheme are given, namely the area for the circuit with CED and the percentage area overhead compared with the optimized functional logic without CED. For the proposed scheme different tradeoffs between area overhead and coverage are shown. The last coverage/overhead entry for each circuit shows the case where no even error detecting circuit is used (i.e., where only single-bit parity is used). To increase coverage, the even errors have to be detected.

With a sufficiently large value of L, 100% coverage was obtained for most circuits to give a reference point. Note that the percentage area overhead was computed as follows:

% overhead = (area with CED – optimized area without CED) / (optimized area without CED) $\times$100

The coverage was computed in the manner described in Sec. 2.4 where faults were randomly injected in the functional logic and random patterns were simulated. The coverage is defined as the number of output vectors that contained errors that were detected by the CED. Of course, duplication always provides 100% coverage.

As can be seen from the results, significant reductions in area overhead can be achieved with relatively small reductions in coverage. It is interesting to note that in most cases, getting the last 1-2% of coverage is very expensive. By going from 100% down to 99-98% coverage, a significant reduction in the CED overhead can be achieved. The likely reason for this is that there are a number of hard to sensitize paths that lead to even errors. Since few patterns sensitize these paths, the probability of soft errors occurring along these paths is very small. However, detecting these soft errors requires a lot of hardware. This phenomenon is illustrated in Figs. 2.4-2.6 which are graphs of coverage versus overhead. As can be seen in these graphs, the CED hardware required to increase the coverage rises somewhat linearly until the coverage reaches the high 90's at which point a lot of hardware is required to detect the last few percent of soft errors. The proposed method provides a very efficient way to take advantage of this phenomenon by allowing the designer the option of reducing the CED overhead significantly with only small loss in coverage.

Table 2.1: Comparison of Proposed Method with Duplication

Circuit				Duplication		Proposed		
Name	Num. PI	Num. PO	Area	Area	Overhead (%)	Area	Overhead (%)	Coverage (%)
apla	10	12	5348	12298	130.2	8252	54.3	100
						7487	40.3	99.2
						6417	20.5	88
						6150	11.5	72.4
br1	12	8	2876	6614	129.5	3756	30.6	100
						3402	18.3	99.8
						3258	13.3	91.2
						3169	10.2	76.4
chkn	29	7	8120	18940	133.5	16516	103.4	99.8
						15687	93.2	98.7
						15509	91.4	96.7
						15395	89.6	90
dc2	8	7	3021	7046	133.2	4894	62.1	100
						3697	22.4	99.4
						3359	11.2	90.3
						3217	6.5	84.6
exp	8	18	4176	9396	131.3	5846	40.4	100
						5203	24.6	97.4
						5094	22.2	80
						4927	18.4	70.2
wim	4	7	1236	3414	176.2	2224	80.2	100
						2007	62.4	98.9
						1928	56.3	80.2
						1903	54.0	74.2
5xp1	7	10	2320	5220	125.0	4461	92.3	100
						4245	83.6	98
						4129	78.4	88
						4036	74.0	81
b12	15	9	1208	3020	150.1	2356	110.2	100
						2391	98.4	97.3
						2379	97.6	78.4
						2343	94.2	72.2
cu	14	11	1180	3068	160.2	2950	150.1	100
						2430	106.4	98.2
						2289	94.3	72
						2230	89.7	64.2
sao2	10	4	2124	4885	129.9	3557	67.5	100
						3241	52.6	97.8
						3160	48.8	82
						3107	46.3	71.3
misex1	8	7	849	1970	132.0	1867	120.2	99.8
						1793.1	112.1	99.2
						1528.4	80.2	84
						1446.2	70.4	76

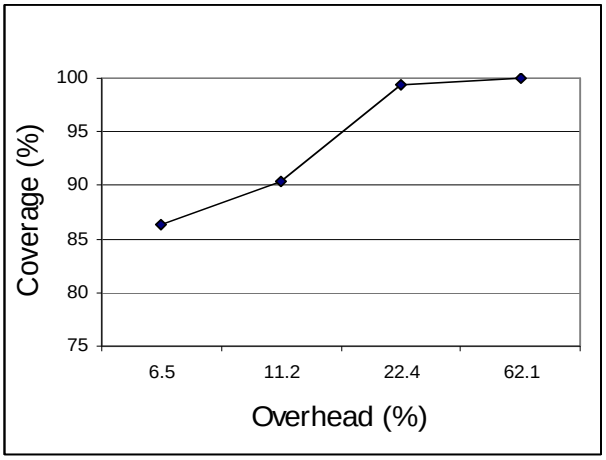

Figure 2.4: Coverage vs. Overhead for *dc2*

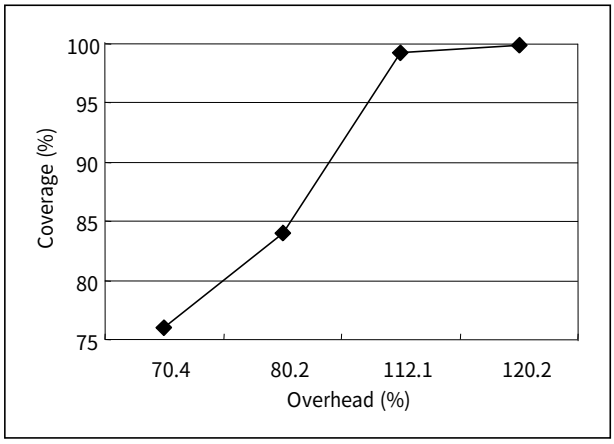

Figure 2.5: Coverage vs. Overhead for *misex1*

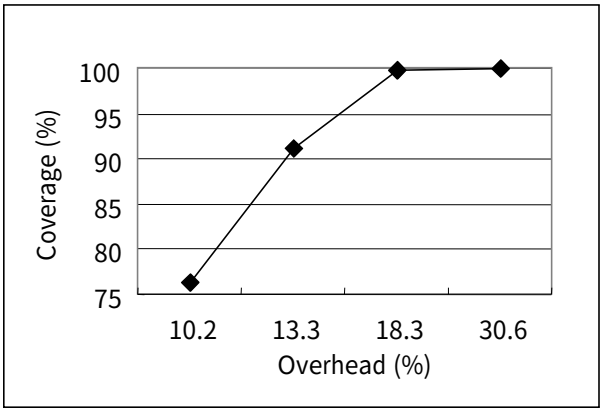

Figure 2.6: Coverage vs. Overhead for *br1*

2.6 CONCLUSION

The proposed method provides an efficient way to achieve high levels of soft error protection with reduced overhead. It is non-intrusive and thus does not require any modification to the functional logic itself.

Chapter 3: Multiple Bit Upset Tolerant Memory Using a Selective Cycle Avoidance Based SEC-DED-DAEC Code

3.1 INTRODUCTION

Ionizing radiation from high-energy neutrons and alpha particles can cause a single-event upset (SEU) that may alter the state of the system resulting in a soft error. Memories, which occupy a large percentage of the area of a chip, are especially sensitive to SEUs. Constant technology process improvement has resulted in very dense memory cells that store information with less capacitance and lower voltage. Consequently, less charge is required to produce one or more soft errors in memories. Recent studies characterizing different bit errors arising from an SEU suggest that 1–5% of the SEUs can cause multiple bit upsets (MBUs) [Maiz 03]. Depending on the underlying technology and the incident particle, several types of multiple-bit errors are possible [Satoh 00], [Makihara 00], [Kawakami 04]. It has been shown that incident neutron particles can react with the die contaminants and generate secondary particles with enough energy to create multiple errors. The distance between the bits in error depends on the initial angle of incidence, die contaminant types, and the scattering angle for the secondary particles. Based on this, the probability of adjacent double bit errors is much higher than other multiple bit errors.

A SEC-DED code [Hamming 50] is capable of correcting one error and detecting all possible double errors. It is commonly used in memories and caches, but cannot correct more than a 1-bit error in a word. In order to correct the most commonly occurring MBUs, this chapter proposes a low cost ECC methodology to correct double adjacent bit errors. It involves constructing a single-error-correcting, double-error-detecting, double-adjacent-error-correcting (SEC-DED-DAEC) code by selectively

avoiding certain types of linear dependencies in the parity check matrix. A key feature of the proposed SEC-DED-DAEC code is that it uses the same number of check bits and has nearly identical syndrome generator/encoder area and timing overhead as the conventional SEC-DED codes. Consequently, there is very little additional cost to using it. Specific *H*-matrices for 16, 32 and 64-bit data words are given in the chapter, and their properties are directly compared with commonly used SEC-DED codes published elsewhere.

While the focus in the chapter is on SEC-DED-DAEC codes, the proposed methodology is flexible and can be used to construct codes for correcting any subset of double errors.

3.2 RELATED WORK

A number of approaches for extending the basic SEC-DED Hamming code [Hamming 50] have been previously proposed. A special class of SEC-DED codes known as Hsiao codes [Hsiao 70] was proposed to improve the speed, cost, and reliability of the decoding logic. The codes constructed in the proposed methodology can be thought of as a special class of Hsiao codes. Another class of SEC-DED codes [Reddy 78], [Chen 83] was proposed to detect any number of errors affecting a single byte. These codes are known as single-error-correcting double-error-detecting single-byte-error-detecting (SEC-DED-SBD) codes. For protecting byte-organized memories, SEC-DED-SBD codes are more suitable than the conventional SEC-DED code.

To provide byte error correction capability, single-byte-error-correcting, double-byte-error-detecting (SBC-DBD) codes [Berlekamp 68], [Reed 60], [Wolf 69], [Bossen 70] [Chen 96] were proposed. These codes perform at a higher order Galois field and consequently the encoding and decoding are more complex. Moreover, they require more check bits thereby increasing the size of the memory.

To provide complete double error correction capability, a double-error-correcting triple-error-detecting (DEC-TED) code may be used at the cost of much larger overhead in terms of both the check bits and more complex hardware to implement the error correction and detection [Lin 83], [Berlekamp 68], [Lala 78].

The Reed-Solomon (RS) code and Bose-Chaudhuri-Hocquenghem (BCH) codes are able to detect and correct multiple bytes of errors with very low overhead in terms of additional check bits required. However, these codes typically work at the block level and are applied to multiple words at a time. Other similar codes include the extended Hamming code [Bossen 70] which performs at a higher order Galois field GF (2^K) and can correct up to *k*-bit burst errors. Other multiple error correcting codes include the optimal rectangular code (ORC), adaptive cross-parity code (APX) code, and others. The general drawbacks with these methods are latency and speed. Most of these codes require several cycles to correct the first error. Moreover, the encoding and decoding are much more complex and require several table lookups for multiplication in higher order fields.

Another class of multiple error-correcting approaches combines coding with circuit level techniques to sense multiple errors in a memory. In [Vargas 94] and [Calin 95], an asynchronous built in current sensor (BICS) on the vertical power lines of a memory along with a parity bit per memory word is used. A conventional SEC-DED code and the BICS approach are combined in [Gill 05] to detect multiple bit upsets affecting the same memory word.

Even though several powerful error correcting codes exist, the SEC-DED code has remained an attractive choice mainly because of its fast and simple encoding/ decoding and low hardware overhead. One of the most commonly used techniques to minimize the probability of multiple bit upsets in a single word is *bit interleaving* [Maiz 04] which is a memory layout architecture in which physically adjacent bits are assigned

to different logical words. For k-way interleaving, k adjacent failing bits appear as k different single-bit errors in k different logical words rather than as a k-bit error in a single logical word. A simple SEC-DED code can be used along with bit interleaving to help protect from multiple bit upsets. However, there can be some limitations/drawbacks for bit interleaving. In some cases, it may negatively impact floorplanning, access time, and/or power consumption. The proposed SEC-DED-DAEC code requires very little overhead and can be used instead of or in addition to bit interleaving to provide greater flexibility for optimizing a memory design. For a fixed depth of interleaving, a larger physical distance between cells in error can be tolerated using the proposed code, or to tolerate a fixed physical distance of cells in error, the required depth of interleaving can be reduced. The proposed methodology places an additional tool in the hands of a memory designer for optimizing a memory layout. Moreover, for small memories, e.g., content addressable memory or register files, interleaving may not be feasible. The proposed coding methodology is particularly useful in this case to provide protection from MBUs.

A class of systematic SEC-DED-DAEC codes was proposed much earlier in [Abramson 59]. However, it was not targeted for memories. Its encoding and decoding are not as efficient as conventional SEC-DEC codes. One check bit is dedicated to differentiate between single and double bit errors. This check bit computes the parity of the entire message and hence incurs a lot of decoding delay and large decoder overhead. Moreover, the encoding and decoding involve the use of a linear finite state machine (LFSM) and hence the latency is increased. Some extensions of the basic code in [Abramson 59] have been suggested. In [Elspas 60], the SEC-DED-DAEC code was extended to higher order fields GF (2^K), and in [Bernstein 63], the code was modified for arithmetic operations.

The ECC methodology proposed in this chapter constructs a different SEC-DED-DAEC code from the ones described in [Abramson 59], [Elspas 60], and [Bernstein 63]. The proposed SEC-DED-DAEC codes are targeted for memories and have the same number of check bits and nearly identical encoding and decoding latency as conventional SEC-DED codes. The proposed codes are constructed by selectively avoiding certain type of cycles in the parity check matrix. Moreover it tries to minimize the miscorrection (non-adjacent double error mistaken as an adjacent double error) probability.

3.3 BINARY LINEAR BLOCK CODES

The proposed SEC-DED-DAEC code falls into the category of systematic binary linear block codes. A binary (n, k) linear block code is a k-dimensional subspace of a binary n-dimensional vector space. An n-bit codeword of the code contains r=(n-k) check bits and k data bits. The ($r \times n$) parity-check matrix (H-matrix) completely defines the code. C is a codeword of the code if and only if

$$H.C^T = 0 \qquad (1)$$

where C^T is the transpose of the codeword C. The H-matrix corresponds to a systematic code if it can be represented as

$$H=[P_{r\times k},I_{r\times r}] \qquad (2)$$

where I is the $r \times r$ identity matrix. For a systematic code, the first k-bits of the codeword can be designated as the data bits and the last r bits can be designated as the check bits. For the targeted application, only systematic codes are useful. For a systematic code with a parity check matrix of the form given by Eqn. 2, the generator matrix can be simply obtained as

$$G=[I_{K\times K},P^T] \qquad (3)$$

The H-matrix represents a set of linear equations involving the bits of the message. The syndrome is defined as the r-bit vector obtained upon multiplying the

received n-bit message with the H-matrix in GF (2). In the error free case, the syndrome is the all-zero vector. An error vector is defined as an r-bit vector where the bits that are in error have the value 1 and all the other bits are 0. An erroneous message V_e can be represented as

$$V_e = V + E \tag{4}$$

where E is the error vector and V is the error free message (i.e., codeword).

$$S = H.V_e = H.(V+E) = H.V + H.E = H.E \tag{5}$$

where S is the syndrome for the particular message V_e. In the next section, we will discuss the proposed linear systematic block code.

3.4 PROPOSED CODE

The proposed SEC-DED-DAEC code has the following properties:

1. All single-bit errors can be corrected
2. All double bit errors can be detected
3. All adjacent double bit errors can be corrected

The miscorrection probability for non-adjacent double errors is reduced

The characteristics of a linear block code are completely determined by its H-matrix. To detect all single-bit errors, the corresponding error syndromes should be unique. Note that the syndrome for a single-bit error at the bit position p is the same as the p-th column of the H-matrix. To uniquely identify all the single-bit errors, all the columns of the H-matrix must be unique.

To detect all the double bit errors, the corresponding syndromes should be different from all the single-bit error syndromes. The syndrome for a double bit error is given by the exclusive-or (XOR) of the corresponding columns of the H-matrix. So there cannot be any 3-cycle in the H-matrix. A k-cycle refers to a set of k linearly dependent

columns of the parity check matrix, i.e., when XORed together, the output is an all-zero column. To be able to correct all the adjacent double bit errors, the syndromes for the adjacent double bit errors should be different from each other and also different from all the single-error syndromes. Next we define the conditions that must be satisfied by the *H*-matrix for the proposed code:

1. No all 0 columns.
2. All columns are distinct
3. No linear dependency involving 3 or less columns i.e., no 2-cycle and 3-cycle are allowed.
4. No linear dependency involving columns C_i,C_j,C_k,C_m where $m>k>j>i$, such that $j=i+1$ and $m=k+l$.

Moreover the code tries to minimize the number of 4-cycles involving C_i,C_j,C_k,C_m where $m>k>j>i$, such that $j=i+1$ or $k=j+l$ or $m=k+1$.

Condition 1 ensures that no single-bit error case match the error free case.

Condition 2 ensures that all the single error syndromes are unique. Every single error syndrome matches one of the columns of the *H*-matrix. Since all the columns of the *H*-matrix are distinct, the single-bit errors are uniquely identifiable and hence correctable. Additionally this condition ensures that there are no pairs of double errors of the form (*i,j*) and (*j,k*) such that the corresponding syndromes are the same. Assume that such double errors exist, then $(C_i \oplus C_j) \oplus (C_j \oplus C_k)=0$, i.e., $(C_i \oplus C_k)=0$ but that contradicts the fact that all the columns of the *H*-matrix are distinct. This ensures that syndromes for adjacent errors of the form $(i,i+1)$ and $(i+1,i+2)$ are different.

Condition 3 ensures that the syndromes for all double bit errors are different from that of the single-bit errors. The syndrome for a double bit error is determined by the XOR of the columns corresponding to the erroneous bit positions. If the *H*-matrix is free

of 3-cycles then the XOR of any two columns of the *H*-matrix is not identical to any of the columns of the *H*-matrix. This ensures that the syndromes of all the double bit errors are different from the single-bit error syndromes, and condition 2 ensures that the double bit error syndromes are non-zero. Hence all the double bit errors are detectable.

Condition 4 along with condition 2, ensures that a syndrome for an adjacent double bit error is different from all other adjacent double bit error syndromes. If we assume that the only errors are single-bit errors or adjacent double bit errors, then with an *H*-matrix satisfying conditions 1 through 4, we can uniquely identify the syndromes for all single-bit errors and adjacent double bit errors and hence can correct all single-bit errors and all double adjacent bit errors and detect all double bit errors.

However the syndromes for the adjacent bit errors are shared with some non-adjacent double bit error syndromes. This is because some 4-cycles are allowed in order to reduce the check-bit overhead. So there is a possibility that a non-adjacent double bit error will be mistaken as an adjacent double bit error and hence will be incorrectly corrected (although the probability of non-adjacent double errors is much less than that of the adjacent double errors). Condition 5 tries to minimize the probability of such an event happening. We call the 4-cycles of the type given by condition 4, *forbidden 4-cycles (4FC)*. We call the 4-cycles of the type C_i,C_j,C_k,C_m where $m>k>j>i$, such that $j=i+1$ or $k=j+1$ or $m=k+1$, *bad cycles (4BC)*, since their presence have a detrimental effect on the capacity of the code. The number of non-adjacent double bit errors is $C_2^n - (n-1)$. For the double errors that are caused by independent SEUs, all the double errors are equally likely. In that case, the miscorrection probability is given by:

$$Pr\ (miscorrection) = \frac{\#4BC}{C_2^n - (n-1)} \qquad (6)$$

While constructing the *H*-matrix, an effort is made to reduce the miscorrection probability by keeping the number of 4BCs small. While designing the *H*-matrix,

additional constraints can be imposed to reduce the encoding and decoding overhead. This can be achieved by limiting the number of 1's in any row of the *H*-matrix.

3.5 CODE DESIGN PROCEDURE

The design of the *H*-matrix is essentially a systematic search process to satisfy all the conditions mentioned in the previous section. For an ($r \times n$) matrix, there are $2^{(r \times n)}$ possible choices, so an exhaustive search approach is ruled out for reasonably large values of *r* and *n*. Figure 3.1 shows a *H*-matrix for a (22,16) code. Even for this code, an exhaustive search is not practical even if the domain of columns considered is restricted. The *weight* of a column of the *H*-matrix is defined as the number of 1's in the column. If we limit the *H*-matrix to only weight-3 and weight-1 columns, then there are $C_3^6 = 20$ choices out of which 16 columns can be chosen in $C_{16}^{20} = 4845$ ways. For each choice there are ($16! > 2 \times 10^{13}$) column permutations which should be searched for the best code. So an exhaustive search will have to search ($4845 \times 16!$) > (2×10^{13}) matrices for the best code. The search space increases further if arbitrary weighted columns are allowed. Note that For the H-matrix in Fig. 3.1, no column of weight two is allowed because any weight-2 column will create a 3-cycle with two weight-1 columns.

Constructing the best *H*-matrix for a SEC-DED-DAEC code that satisfies all of the conditions discussed in Sec. 3.4 is NP-complete. A pseudo-greedy search procedure can be used as shown in Fig. 3.3. The outer while loop stops once a valid code is found or the maximum backtrack limit is exceeded. The inner while loop finds a set of valid columns (that does not introduce any forbidden cycles) for the current column position. If no valid column is found for the current column position, then the last choice for a column has to be undone. This corresponds to a backtrack. If multiple valid columns are found for the current column position then the one that minimizes the number of bad 4-cycles in the currently constructed code space is chosen. Once an initial *H*-matrix is

found, a limited number of column permutations are tried to avoid a local optimum and search for a better *H*-matrix in terms of reduced miscorrection probability.

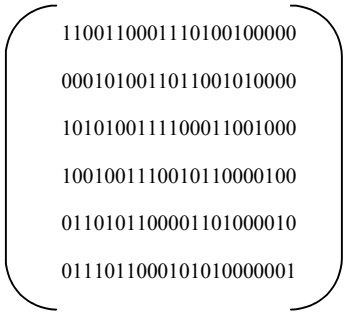

Figure 3.1: *H*-matrix for proposed (22,16) code

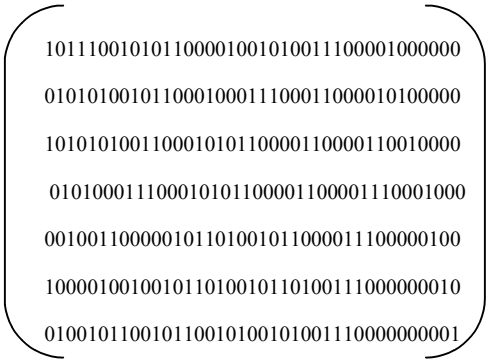

Figure 3.2: *H*-matrix for proposed (39,32) code

Input: *n*, *maxIter, maxBacktrack, maxPermute*

Output: *H*-matrix

avail_col = All 1-weight columns, followed by 3-weight columns, followed by 5-weight columns, and so forth up to the largest weight columns being considered

currentCol = 0; *backtrack* = 0

while (*currentCol* < *n*) {

Iter = 0

```
    validColPool[currentCol] = {}
    while ( iter < maxIter ) {
        Iter++
        C = Untried least-weighted column from avail_col
        Check for existence of forbidden 4-cycles
        if ( ! 4FCfound ) {
            validColPool[currentCol] =
                    validColPool[currentCol] ∪ C          }      }
    if ( empty(validColPool[currentCol]) ) {
        backtrack++
        if ( backtrack > maxBacktrack ) {
            return      // no code found
        } else {
            currentCol--
            if ( currentCol < 0 )   currentCol = 0;
            continue;
        }   } else {
        sCol = selectMin4BC(validColPool[currentCol])
        add sCol to H-matrix
        currentCol++
        backtrack=0;      }}
permuteC = 0;   orig4BC=count4BC(H-matrix);
while ( permuteC < maxPermute) {
    permuteC++
    permuteColumns()
    Check for existence of forbidden 4-cycles
    if ( (!4FC)&&(count4BC(H-matrix)<orig4BC)) {
```

H-matrix ← current H-matrix; }}

Figure 3.3: Pseudo-greedy search algorithm

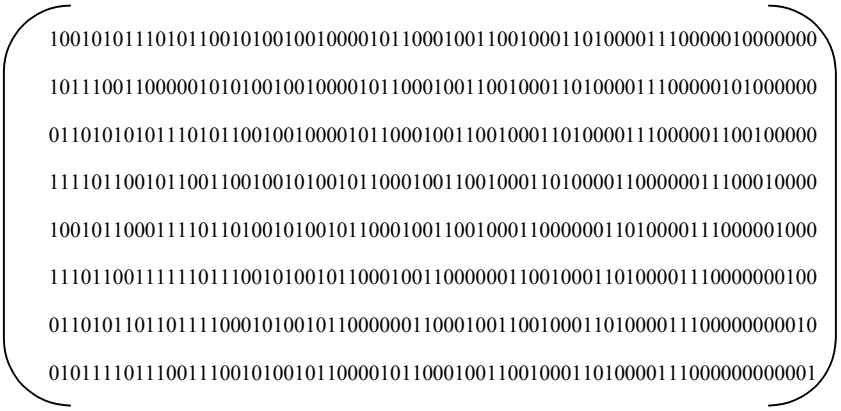

Figure 3.4: *H*-matrix for proposed (72,64) code

Table 3.1: Comparison of proposed SEC-DED-DEAC code with other codes

(n,k)	Codes	2-Input XOR Gates	Max Logic Depth	Forbidden 4-Cycles (4FC)	Total 4-Cycles	Bad 4-Cycles (4BC)
(22,16)	SEC-DED (IBM system/3)	48	4	13	252	122
	Hsiao Code [Hsiao 70]	48	4	8	252	120
	SEC-DED-DAEC in [Abramson 59]	70	5	0	252	128
	Proposed SEC-DEC-DAEC (Fig. 3.1)	48	4	0	251	118
(39,32)	SEC-DED (IBM 8130)(40,32)	96	4	82	776	254
	Hsiao Code [Hsiao 70]	96	4	23	1363	425
	SEC-DED-DAEC in [Abramson 59]	132	6	0	1343	386
	Proposed SEC-DED-DAEC (Fig. 3.2)	96	4	0	1363	379
(72,64)	SEC-DED (IBM 3081)	256	6	230	8912	1292
	Hsiao Code [Hsiao 70]	208	5	122	8392	1399
	SEC-DED-DAEC in [Abramson 59]	296	7	0	8194	1335
	Proposed SEC-DED-DAEC (Fig. 3.4)	224	5	0	8289	1316

The number of the 2-input XOR gates required for the encoding/decoding can be computed from the *H*-matrix. It is equal to $\sum_{\#rows}$ *(row weight − 1)*. The encoding and decoding delays are determined by the maximum logic-depth of the encoder and the decoder circuit which is equal to log_2 *(max. 1's in any row)*. Figure 3.2 shows an *H*-matrix for the (39,32) code constructed using the search process as discussed above using only weight-1 and weight-3 columns. Another *H*-matrix is shown for a (72,64) code in Fig. 3.4. In this case, weight-1, weight-3, and weight-5 columns are used.

Table 3.1 shows the number of XOR gates and maximum logic depth for the syndrome generator, number of forbidden 4-cycles, total number of 4-cycles, and the number of bad cycles (4BCs) for the (22, 16), (39, 32) and (72,64) codes for both the proposed code and the SEC-DED-DAEC code described in [Abramson 59] as well as some Hsiao codes and SEC-DED codes commonly used in industry. Note that the SEC-DED code and the Hsiao code cannot correct adjacent double bit errors because of the existence of forbidden cycles (4FCs). The check-bit overhead for a random double error correcting code (DEC) is unacceptably high. For example to protect a 32-bit word, a DEC needs at least 11 check bits. To protect a 64-bit word, a DEC code needs 14 check bits. Using a DEC code can increase the memory size considerably and hence is not an attractive choice for memory ECC. The XOR gate overhead for the proposed code is similar to that of the Hsiao code. The proposed code also has minimal logic depth among the codes and also minimum check bit overhead. The total number of bad 4-cycles is lower for the proposed code than for [Abramson 59], and consequently it has a lower miscorrection probability.

3.6 ECODING /DECODING ALGORITHM

The proposed code is systematic. During encoding, the data bits can be directly copied and the check bits are generated using an XOR network corresponding to the *G*-matrix. The decoding algorithm is as follows:

1. Generate the syndrome using an XOR network corresponding to the *H*-matrix.
2. If the syndrome is the all zero vector, then no error is detected, otherwise one or more errors occurred.
3. If the syndrome matches any of the *H*-matrix columns, then a single error is detected and the error position is the corresponding column position. The corresponding bit should be flipped to correct the error.
4. Else if the syndrome matches any of the *n-1* adjacent double error syndromes, then a double adjacent error is detected and the corresponding bit positions are generated using the error correction logic.
5. Else an uncorrectable error (UE) (i.e., a double non-adjacent error or more than two errors) has occurred.

The only additional overhead with respect to a conventional SEC-DED code comes from step 4 of the decoding step. Figure 3.5 shows the basic error detection and correction block diagram. If a non-zero syndrome is encountered, then the OR gate flags an error indication. If the syndrome matches any of the single error syndromes then the syndrome decoder generates a 1 in the erroneous bit position. Otherwise, if the syndrome matches any of the adjacent double error syndromes, then the decoder generates 1's at the erroneous adjacent bit positions. Otherwise the output of the syndrome decoder is the all zero output. The syndrome decoder consists of 3-input OR gates whose inputs are driven by outputs of *r*-input AND gates. The *i*-th output of the decoder is 1 if and only if a single

error occurred at the *i*-th bit or a double-adjacent error occurred at <*i*,*i*+1> bits or <*i*-1,*i*> bits. The outputs of the decoder are used to generate the corrected word, by using *n* 2-input XOR gates. If the syndrome is non-zero and does not match any of the single or double-adjacent error syndromes, then an uncorrectable error (UE) is encounter and the UE signal is flagged.

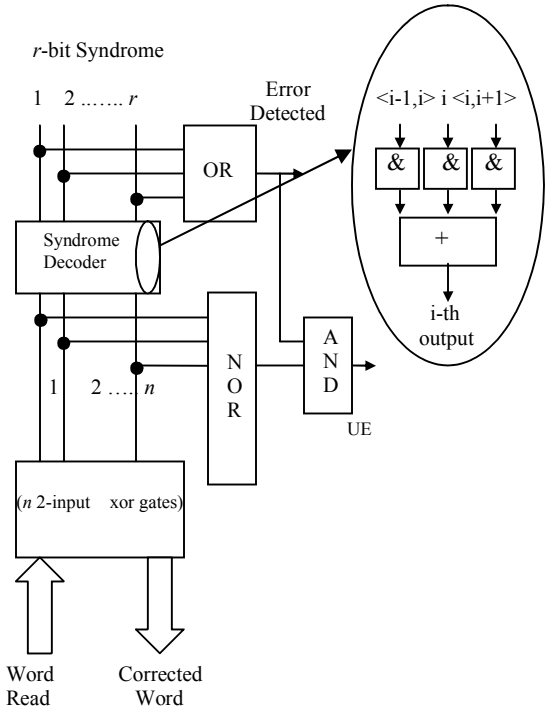

Figure 3.5: Error detection and correction block diagram

3.7 CONCLUSIONS

The ECC methodology described in this chapter adds the ability to correct adjacent errors at very little cost over conventional SEC-DED codes. The only drawback is the possibility of miscorrection for a small subset of multiple errors, however MBUs caused by a single SEU have a much higher probability of occurring than having multiple

independent SEUs accumulating in the same word (this is especially the case if memory scrubbing is used). While bit interleaving is commonly relied upon to protect memories from MBUs, the proposed methodology provides another tool for a memory designer to use. In some instances, it may be an attractive alternative to bit interleaving to allow for a more optimized memory layout, or it can be used in addition to bit interleaving to provide an additional layer of protection from MBUs.

Chapter 4: Multiple Bit Upset Tolerant Router Memory Using a Low Cost Unequal Error Protection Code

The network-on-chip (NoC) paradigm is seen as a way of facilitating the integration of a large number of computational and storage blocks on a chip to meet several performance and power constraints. However due to continued scaling of process technologies, the devices and interconnects have become more sensitive to reliability threats such as, single event upsets and crosstalk. This chapter presents a low cost error correcting code based technique to protect the NoC routers against single event upset induced soft errors and also against crosstalk. An unequal protection error correcting code based methodology is provided for the most commonly used store-and-forward routing strategy. The proposed code has the same check bit overhead as the conventional single error correcting (SEC) code. The encoding/decoding overhead and latency are also similar to the conventional low cost SEC code. The proposed codes belong to the class of unequal error protection codes as they provide different levels of error correction capability for different portions of the same packet with more protection for the important parts of the data.

4.1 INTRODUCTION

The advent of nanometer technologies has facilitated huge amount of transistors in a single die. Reduced feature sizes along with increasing transistor densities have transformed the on-chip interconnect into the deciding factor in meeting the performance and power consumption budgets of a design. Several interconnection schemes are currently in use, including crossbars, rings, buses, and network-on-chip (NoC's) [Krewell 05]. The bus and NoC based architectures are the most prominent and have been widely

studied in the research community. However, buses suffer from poor scalability. As the number of cores increases, the performance of bus based architectures degrades dramatically. This has led to increased adoption of packet based interconnection networks known as network-on-chip. The NoC architectures offer a variety of advantages. A well designed NoC uses wires more efficiently and uses the same wires for multiple purposes. The reduced requirement of global wires improves power dissipation, signal integrity, less silicon area and better physical routability. The NoC topology can be tuned to the application leading to little arbitration, less wait states, and lower power utilization than a bus. The packet based architecture provides better scalability.

For NoC, the underlying network must meet quality of service requirements (such as reliability, guaranteed bandwidth/latency), and deliver energy efficiency [Pande 06]. And all these should be achieved under the limitation of intrinsically unreliable signal transmission media. These limitations are caused by the increased likelihood of timing and data errors [Rossi 05], the variability of process parameters, crosstalk and environmental factors e.g., electro-magnetic interference (EMI) and soft errors. The increased sensitivity to soft errors is caused by the reduction in transistor dimensions and the reduction of supply voltage. Ionizing radiation from high-energy neutrons and alpha particles can cause a *single-event upset (SEU)* [Nicolaidis 05] that may alter the state of the system resulting in a *soft error.* Radiation induced single event upsets can cause bit-flips in the sequential logic elements such as the router buffers, memories, registers [Kastensmidt 05].

Some work has been done to protect on-chip interconnects against crosstalk [Rossi 05], [Nieuwland 05-2], [Bertozzi 02]. In [Nicolaidis 05], [Kastensmidt 05], [Bertozzi 02], and [Lajolo 01], several techniques were proposed to protect the on-chip

sequential elements against SEUs under the assumption that the links are not affected by soft errors. For NoC architectures both the links as well as the router buffers can be affected by soft errors. In [Frantz 06], a new technique was proposed that can simultaneously deal with SEU and crosstalk effects in the NoC routers. A combination of error correcting codes (ECC) and hardware and time redundancy were used for this purpose. In [Park 06], another method was proposed to address simultaneously the problem of link errors due to crosstalk, capacitive loading, and SEUs in router buffers. A flit-based hop-by-hop retransmission scheme and corresponding retransmission architecture were proposed.

The routing mode influences the buffer size needed in the routers and the performance of the system, e.g., packet latency. In packet switching networks data items have to be buffered at each router before they are sent over. There are two basic types of routing modes commonly used in NoC architectures: store-and-forward routing and wormhole routing.

In wormhole routing, messages are sent as worms. The packet is split into flits and the flits are sent in contiguous fashion. The first flit contains the destination address and it reserves the channel through which the subsequent flits are sent. This routing architecture facilitates smaller router buffers but this routing strategy may lead to high data contention and consequently lead to higher message latency. The rest of the chapter focuses only on store-and-forward type of routing.

In store-and-forward routing, the entire packet is stored in the router buffer before it can be forwarded to the next router or destination core. In this chapter, we propose an unequal error protection code to protect data packets from SEUs and crosstalk induced link errors. Recent studies characterizing different bit errors arising from an SEU suggest

that 1–5% of the SEUs can cause multiple bit upsets (MBUs) [Maiz 03]. Recent studies show that the most likely multiple-bit-upsets (MBUs) are adjacent double bit errors [Satoh 00], [Makihara 00], [Kawakami 04]. Moreover crosstalk induced link errors also tend to be either single-bit error or double adjacent bit error. Since the life span of a packet in the buffer is small likelihood of the same packet being affected by multiple SEUs is negligible. This eliminates the concern of a packet having random double-bit upsets. The conventionally used SEC codes can detect and correct only single-bit errors and hence can result in data loss in the presence of single event induced adjacent double-bit errors. A single error correcting, double error detecting (SEC-DED) code can be used to provide additional double-bit error detection capability but at the cost of increased check bit overhead, additional encoding and decoding overhead, and latency. However even an SEC-DED code can only correct single-bit errors. If an adjacent double-bit error occurs in the header portion of a packet then even if the error is detected it cannot be corrected and hence the source or the destination address cannot be decoded. This leads to data loss.

In this chapter we propose a single error correcting, double adjacent error detecting, selective double adjacent error correcting (SEC-DAED-SDAEC) code which corrects all single-bit errors and detects all double adjacent bit errors in the whole packet. Additionally it provides adjacent double-bit error correction capability in the header portion. This helps in recovering the header portion of the packet in the presence of adjacent double errors in the header and hence a retransmission request can be sent. This will prevent data loss due to single event induced or crosstalk induced adjacent double-bit errors. An attractive feature of the proposed code is that it has the same check bit

overhead, encoding overhead, and latency, as an SEC code. The decoding is slightly more complex because of the additional error detection and correction capability.

The rest of the chapter is organized as follows. In Sec. 4.2, the basic architecture of a NoC router is described. Sections 4.3, 4.4, and 4.5 discuss the proposed coding scheme for the store-and-forward type of routing and also describe the code design procedure. Comparison of the proposed code with all the relevant existing codes is also provided. In section 4.6, the encoding and decoding algorithms for the proposed code are described. Section 4.7 concludes the chapter.

4.2 NOC ROUTER ARCHITECTURE

The most active component of the network is the router and hence is key to achieving reliability and performance standards. Figure 4.1 shows the basic outline of a NoC router. The core of a router basically consists of 1) input ports to receive packets and store them in buffers and in some architectures can also send acknowledgement signals, 2) output ports which receive packets from an input port of the same router and send them to input ports of another router or destination core and can also receive acknowledgement signals, 3) routing arbitration logic that decides the destination of packet 4) switch architecture which links the input and output buffers. The router can also be equipped with logic to send and receive retransmission requests upon detection of unrecoverable errors. The packet while residing in the router buffer can be affected by single event upsets. The packet can also be affected by link errors resulting from crosstalk, coupling noise, and transient faults during transmission.

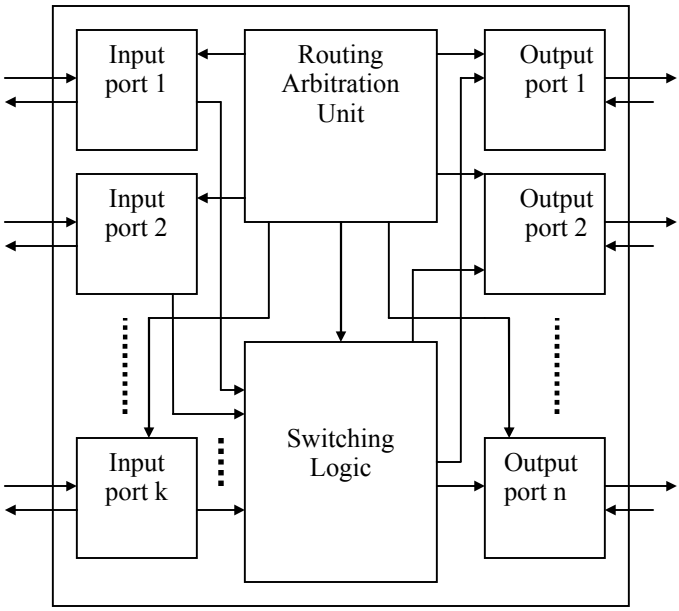

Figure 4.1: Basic router architecture

4.3 STORE-AND-FORWARD ROUTING

In store-and-forward routing the entire packet is stored in the router buffer. The time spent by a packet inside the router primarily depends on three operations: packet buffering, negotiation of an output port, and the transmission to the output. The time spent in negotiation and transmission depends on the contention of the selected output port and on the availability of the outside link. During the lifetime of the packet inside the router buffer, the data can be corrupted due to an SEU which can cause either a single-bit flip or double adjacent bit flips. The data can also be corrupted during transmission from one link to the next. This could be due to crosstalk, coupling noise, or transient faults. The proposed error correcting code encodes the data before it is stored in the router buffer. Next it is decoded to enable router arbitration logic to fetch the destination address and port-id for the packet. Next the data is again encoded before it is transmitted

to the next link. This allows protection against link failures. The basic ECC scheme is shown in Fig. 4.2. In Fig. 2, the "D + E" block refers to the decoding and encoding of the incoming packet. This stage addresses link error induced data corruption. The encoded data resides in the router buffer, and it is decoded before being sent to the router arbitration unit. Finally the data is again encoded before being transmitted to the next link.

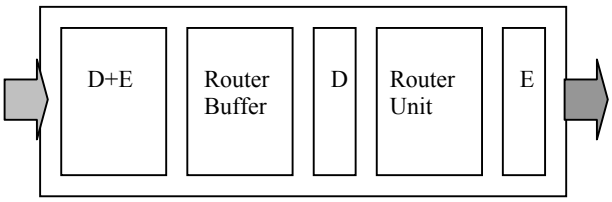

Figure 4.2: ECC scheme for NoC router

Note that conventional SEC or SEC-DED codes are not sufficient to protect data packets against SEU induced adjacent double-bit errors. This is because an uncorrectable error in the header of the packet will lead to packet loss. In [Bodnar 03], a single-error-correcting, double-adjacent-error detecting within a byte, code was proposed which can correct all single-bit errors in the entire word and also detects all double adjacent errors within 8-bit nibbles. But this code is not sufficient in the NoC environment as it cannot correct adjacent double errors in the header part and also cannot detect double adjacent bit errors in the nibble boundaries within a word.

4.4 UNEQUAL ERROR PROTECTION CODE

Unequal error protection codes (also known as unequal protection codes) were first proposed by Masnick and Wolf [Masnick 67]. Later several variants of this type of

code were proposed [Morelos-Zaragoza 94], [Hayashi 00]. These codes provide different levels of protection to different bits of the same word. This is achieved by conditioning the linear dependencies in the parity check matrix (H-matrix) of the code. In general, these codes have the property that some of the digits in a codeword will be decoded correctly only if J_2 or fewer errors occur and others will be decoded correctly only if J_1 or fewer errors occur where $J_1 > J_2$. In [Fujiwara 98], several two-level UEP codes were proposed. These codes were designed as to provide *b*-bit burst error correction capability in one *b*-bit portion of the codeword and either SEC or SEC-DED capability in the remaining portion of the codeword. Another construction of these types of codes was proposed in [Namba 03].

However these codes incur a lot of check bit overhead as well as encoding/decoding overhead and latency. Hence these codes are not suitable for protecting data packets residing in the router buffer in NoC designs. Next we describe the proposed low cost unequal error protection code to protect data packets against SEUs in the router buffer as well as a link error during transmission.

4.5 PROPOSED CODE

The coding schemes proposed in this chapter fall into the category of systematic binary linear block codes. A binary (n, k) linear block code is a k-dimensional subspace of a binary n-dimensional vector space. An n-bit codeword of the code contains $r=(n-k)$ check bits and k data bits. The ($r \times n$) parity-check matrix (H-matrix) completely defines the code. C is a codeword of the code if and only if

$$H.C^T = 0 \qquad (1)$$

where C^T is the transpose of the codeword C. The H-matrix corresponds to a systematic code if it can be represented as

$$H=[P_{r\times k},I_{r\times r}] \qquad (2)$$

where I is the $r\times r$ identity matrix. For a systematic code, the first k-bits of the codeword can be designated as the data bits and the last r bits can be designated as the check bits. For the targeted application, only systematic codes are useful. For a systematic code with a parity check matrix of the form given by Eqn. 2, the generator matrix can be simply obtained as

$$G=[I_{K\times K},P^T] \qquad (3)$$

The H-matrix represents a set of linear equations involving the bits of the message. The syndrome is defined as the r-bit vector obtained upon multiplying the received n-bit message with the H-matrix in GF (2). In the error free case, the syndrome is the all-zero vector. An error vector is defined as an r-bit vector where the bits that are in error have the value 1 and all the other bits are 0. An erroneous message V_e can be represented as

$$V_e = V + E \qquad (4)$$

where E is the error vector and V is the error free message (i.e., codeword).

$$S = H.V_e = H.(V+E) = H.V + H.E = H.E \qquad (5)$$

where S is the syndrome for the particular message V_e. Next we will discuss the proposed unequal error protection linear systematic block codes. The proposed single-error-correcting, double-adjacent-error-detecting, selective-double-adjacent-error-correcting (SEC-DAED-SDAEC) code has the following properties:

- All single-bit errors can be corrected
- All double adjacent bit errors can be detected
- All adjacent double-bit errors in the header of the packet and the one at the intersection of the header and data part can be corrected

In a store-and-forward routing scheme, a data packet typically has a header portion followed by a data section which contains both the data and the check bits. Some architectures assume an additional trailer portion at the end of the packet. In our discussion we are assuming that the header itself contains the information contained in the trailer as well. From a coding perspective it does not make any difference whether the header and the trailer are separated or together as long as they have the same level of error protection. Figure 4.3 shows the basic layout of a packet in the context of the store-and-forward routing.

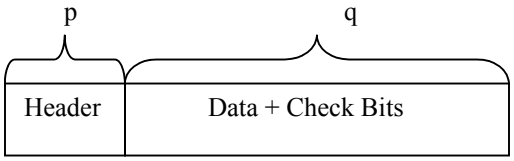

Figure 4.3: Packet structure for store-and-forward routing

For a $(p+q,k)$ SEC-DAED-SDAEC code, the codeword length $n = p+q$ and message length is k, and the number of check bits $r = p+q-k$. An upper bound on the maximum possible codeword length can be obtained for the proposed code as follows:

$$2^r - 1 \geq 2p + q$$

$$2^r - 1 \geq p + n$$

$$n \leq 2^r - 1 - p \qquad (1)$$

$$r = \lceil \log_2(n+p+1) \rceil \qquad (2)$$

Equation 1 is derived from the fact that the least number of unique syndromes required for the proposed code is $(n+p)$, n for the single-bit errors and p for the double adjacent errors in the header. Note that this is a lower bound on the number of unique syndromes required for the proposed code. If the number of check bits is r then the

maximum number of unique syndromes is $2^r - 1$. This number should be more than or equal to the maximum number of unique syndromes required for the code. The characteristics of a linear block code are completely determined by its *H*-matrix. Table 4.1 shows the check bit requirement for different header and data size. The proposed codes are optimal in the sense that the number of check bits used are minimum possible.

Table 4.1: Check bit requirements

Header, data, check bit	Bound on *r*	Min *r*
8,24,r	$r = \lceil \log_2(32 + r + 9) \rceil$	*6*
8,56,r	$r = \lceil \log_2(64 + r + 9) \rceil$	*7*
16,56,r	$r = \lceil \log_2(64 + r + 17) \rceil$	*7*

For the proposed systematic binary linear unequal error protection block code, the H-matrix can be viewed as follows:

$$H = [H_1 \mid H_2 \mid I] \qquad (3)$$

Where H_1 is a $r \times p$ sub-matrix, H_2 is a $r \times (q-r)$ sub-matrix, and I is a $r \times r$ identity matrix.

To detect and correct all single-bit errors, the corresponding error syndromes should be unique. Note that the syndrome for a single-bit error at the *i*-th bit position is the same as the *i*-th column of the *H*-matrix. To uniquely identify all the single-bit errors, all the columns of the *H*-matrix must be unique.

To detect all the adjacent double-bit errors, the corresponding syndromes should be different from all the single-bit error syndromes. The syndrome for a double-bit error is given by the exclusive-or (XOR) of the corresponding columns of the *H*-matrix. So there cannot be any 3-cycle involving adjacent columns in the *H*-matrix. A *k*-cycle refers

to a set of k linearly dependent columns of the parity check matrix, i.e., when XOR-ed together, the output is an all-zero column. To be able to correct all the adjacent double-bit errors in the header portion (H_1 sub-matrix), the syndromes for the adjacent double-bit errors should be different from each other and also different from all the single-error syndromes as well as from all the double adjacent error syndromes in the H_2 part. Next we define the conditions that must be satisfied by the H-matrix for the proposed code:

1. No all 0 columns.
2. All columns are distinct.
3. No linear dependency involving columns C_i, C_j, C_k where $k>j>i$, such that $j=i+1$ or $k=j+1$ or both.
4. No linear dependency involving columns C_i, C_j, C_k, C_m where $m>k>j>i$ and $j \leq p+1$, such that $j=i+1$ and $m=k+1$. This condition implies that the double adjacent error syndromes in the header portion are unique.

Condition 1 ensures that no single-bit error case matches the error-free case.

Condition 2 ensures that all the single error syndromes are unique. Every single error syndrome matches one of the columns of the H-matrix. Since all the columns of the H-matrix are distinct, the single-bit errors are uniquely identifiable and hence correctable. Additionally, this condition ensures that there are no pairs of double errors of the form (i,j) and (j,k) such that the corresponding syndromes are the same. Assume that such double errors exist, then $(C_i \oplus C_j) \oplus (C_j \oplus C_k)=0$, i.e., $(C_i \oplus C_k)=0$ but that contradicts the fact that all the columns of the H-matrix are distinct. This ensures that syndromes for adjacent errors of the form $(i,i+1)$ and $(i+1,i+2)$ are different.

Condition 3 ensures that the syndromes for all adjacent double-bit errors are different from that of the single-bit errors. Hence all the adjacent double-bit errors can be detected.

Condition 4 along with condition 2, ensures that a syndrome for an adjacent double-bit error in the header portion is different from all other adjacent double-bit error syndromes. If we assume that the only errors are single-bit errors or adjacent double-bit errors then with an *H*-matrix satisfying conditions 1 through 4, we can uniquely identify the syndromes for all single-bit errors and adjacent double-bit errors in the header portion. Hence we can correct all single-bit errors and detect all adjacent double errors in the whole codeword. Additionally all double adjacent bit errors in the header can be corrected.

Figure 4.4 shows different possible errors in a codeword and also shows the error detection/correction capability of the proposed code with respect to those errors. Note that the non-adjacent errors may not always be detectable as they might alias with single-bit error or double adjacent bit error and hence may lead to miscorrection. However the probability of non-adjacent error is negligible.

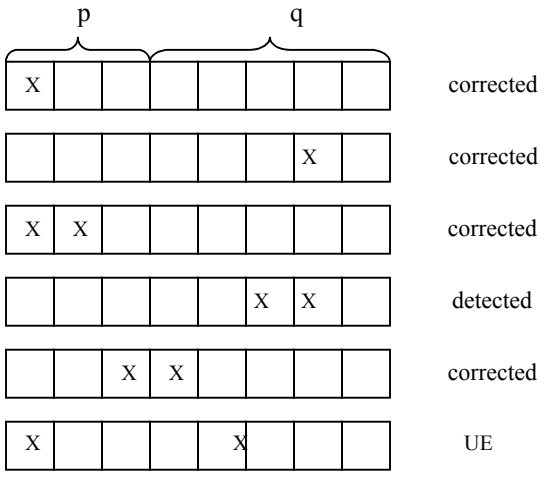

Figure 4.4: Error profiles

Note that in our discussion we are considering only SEU induced soft errors and hence the only possible errors are either a single-bit error or an adjacent double-bit error. This ensures correct decoding of the header portion in the presence of such an error. Since the adjacent error syndromes in the header portion may be shared with some non-adjacent error syndromes, there is a non-zero miscorrection probability. However this probability is negligible because likelihood of the same data packet being affected by non-adjacent double error is negligible.

We call the 3-cycles of the type given by condition 3, *forbidden 3-cycles* (*3FC*). We call the 4-cycles of the type given by condition 4, *forbidden 4-cycles (4FC)*. While designing the *H*-matrix, additional constraints can be imposed to reduce the encoding and decoding overhead. This can be achieved by limiting the number of 1's in any row and column of the *H*-matrix.

4.6 CODE DESIGN PROCEDURE

The *H*-matrix consists of three parts H_1, H_2, I as shown in Eq. 3. The I submatrix is the diagonal identity matrix consisting of all weight-1 columns.

Figure 4.5 shows the outline of the algorithm used to construct the H_2 submatrix. All the columns of the H_2 matrix should be unique. While adding any new column forbidden 3-cycles (3FC) cannot be allowed as that will lead to an aliasing of a single error with a double adjacent error in the data part. At the same time, the algorithm tries to maximize the sharing of double adjacent error syndromes. This allows in reducing the number of used up syndromes and leaves more flexibility while designing the H_1 matrix.

The algorithm maintains a list of syndromes for the single and the double adjacent errors in the constructed code space. A column is a candidate for the next position as long as it does not introduce a 3FC. From a list of candidates, the one is chosen that minimizes the number of double adjacent error syndromes in the constructed code space.

```
Input:   n(codeword length), maxIter, maxBacktrack, r(number of check bits),
p(header size)
Output:  H-matrix
avail_col = Set of all non-zero columns of weight > 1
usedSyndromePool = {}
currentCol = r(starts after identity matrix I);     backtrack = 0
while ( currentCol < n-p ) {
    Iter = 0
    validColPool[currentCol] = {}
    while ( iter < maxIter ) {
        Iter++
        C = An untried column from avail_col
        Check for existence of forbidden 3-cycles
        if ( ! 3FCfound ) {
            validColPool[currentCol] = validColPool[currentCol] ∪ C
        }
    }
    if ( empty(validColPool[currentCol]) ) {
        backtrack++
        if ( backtrack > maxBacktrack ) {
            return      // no code found
        } else {
```

```
                currentCol--
                if ( currentCol < 0 )    currentCol = 0;
                continue;
            }
        } else {
  sCol = selectMinimizeSyndromeUsage(ColPool[currentCol])
  add sCol to H-matrix
  add sCol and adjacent double error syndrome corresponding to sCol to
usedSyndromePool.
            currentCol++
            backtrack=0;
    }
}
```

Figure 4.5: Algorithm to construct H_2

```
Input:  n(codeword length), maxIter, maxBacktrack, r(number of check bits),
p(header size)
Output:  H-matrix
avail_col = Set of all non-zero columns not present in I, H2,
usedSyndromePool;
currentCol = n-p;
backtrack = 0
while ( currentCol < n ) {
    Iter = 0
    validColPool[currentCol] = {}
    while ( iter < maxIter ) {
        Iter++
        C = An untried column from avail_col
```

```
        Check for existence of forbidden 3-cycles and forbidden 4-cycles
        if (( ! 3FCfound )&&(! 4FCfound)) {
            validColPool[currentCol] = validColPool[currentCol] ∪ C
        }
    }
    if ( empty(validColPool[currentCol]) ) {
        backtrack++
        if ( backtrack > maxBacktrack ) {
            return      // no code found
        } else {
            currentCol--
            if ( currentCol < 0 )   currentCol = 0;
            continue;
        }
    } else {
 sCol = selectRandomColumn(validColPool[currentCol])
 add sCol to H-matrix
 add sCol and adjacent double error syndrome corresponding to sCol to
usedSyndromePool.
        currentCol++
        backtrack=0;
          } }
```

Figure 4.6: Algorithm to construct H_1

Figure 4.6 shows the algorithm used to construct submatrix H_1. Here whenever a column is added, a check is made that it does not introduce any forbidden 3-cycle (3FC)

and forbidden 4-cycles (4FC). It ensures that every single error and double adjacent errors in the header portion have unique syndrome and hence correctable.

Figure 4.7 shows a *H*-matrix for a (8, 24, 6) header, data, check-bit) code. Here the only double adjacent error syndromes used for *I* and H_2 are (110000) and all its circular shifts, (100111), (011101), (101100), (001111) i.e.,10 distinct syndromes out of 29 (worst case) possibilities. This is achieved by minimizing the number of unique double adjacent error syndromes in the H_2 portion. This leaves more flexibility while searching for the H_1 matrix. After designing *I* and H_2, the number of available columns for the columns of H_1 and its adjacent error syndromes is (63-30-10) = 23 out of which 8 columns and 8 adjacent error syndromes have to be chosen while avoiding 3FCs and 4FCs.

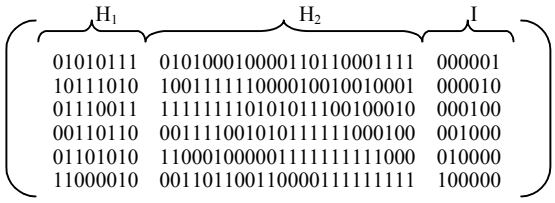

Figure 4.7: *H*-matrix for proposed (8,24,6) code

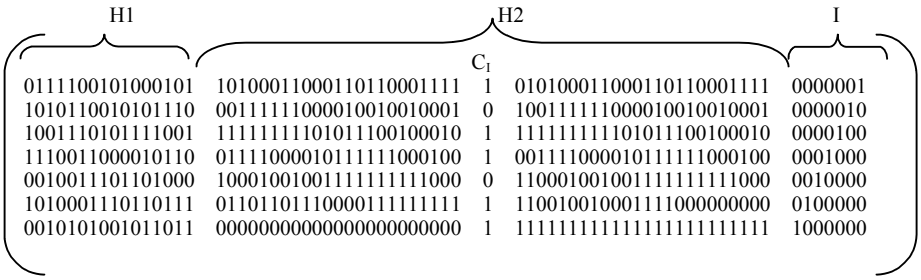

Figure 4.8: *H*-matrix for proposed (16,48,7) code

Table 4.2: Comparison of proposed SEC-DAED-SDEAC code with other codes

Header + Data	Codes	2-Input XOR Gates	Max Logic Depth	Forbidden 3-cycles (3FC)	Forbidden 4-cycles (4FC)	#Check bits
8+24	SEC-DBED [Bodnar 03]	100	5	3	40	6
	SEC	114	5	30	143	6
	$(B_2EC)_8$-$(SEC)_{24}$ [Namba 01] (non-systematic)	98	5	15	0	6
	DEC	-	-	0	0	11
	DAEC [Abramson 59]	132	6	0	0	7
	Proposed SEC-DAED-SDAEC Code	104	5	0	0	6
16+48	SEC-DBED (extended)	231	6	0	263	7
	SEC	204	6	68	316	7
	$(B_2EC)_8$-$(SEC)_{24}$ [Namba 01] (non-systematic)	222	6	27	0	7
	DEC	-	-	0	0	14
	DAEC [Abramson 59]	296	7	0	0	8
	Proposed SEC-DAED-SDAEC Code	240	6	0	0	7

Figure 4.8 shows the *H*-matrix for the (16,48,7) code. Note that the H_2 matrix of a higher dimensional code can be constructed from the H_2 matrix of a lower dimensional code. This can save considerable amount of search time. We illustrate this with the example of the H_2 matrix for the (16,48,7) code which can be obtained from the H_2 matrix of the (8,24,6) code. The H_2 matrix of the (8,24,6) code is a 6×24 matrix which is free of 3FC. In fig. 4.8, ${}^{32}H_2$ denotes the H_2 matrix for the (8,24,6) code where each R_i denotes each row of the matrix. The ${}^{32}H^c{}_2$ denotes the same matrix as ${}^{32}H_2$ except that the last row is complemented. An all zero row is added to ${}^{32}H_2$ and an all one row is added to ${}^{32}H^c{}_2$. A column vector C_1 is introduced at the boundary of ${}^{32}H_2$ and ${}^{32}H^c{}_2$ to avoid any 3FC at the boundary. The resultant H_2 matrix is free of 3FC by construction. Note that the submatrix constructed this way has one extra column. Any column can be removed as

long as no 3FC is introduced. The easiest choice for removal is the leftmost column. This method can be generalized to construct the H_2 matrix of any higher dimensional code from a lower dimensional code. The number of the 2-input XOR gates required for the encoding/decoding can be computed from the *H*-matrix. It is equal to ∑#rows (row weight − 1). The encoding and decoding delays are determined by the maximum logic-depth of the encoder and the decoder circuit which is equal to log_2 *(max. 1's in any row)*.

$$^{32}H_2 = \begin{pmatrix} R1 \\ R2 \\ R3 \\ R4 \\ R5 \\ R6 \end{pmatrix} \qquad ^{32}H^C_2 = \begin{pmatrix} R1 \\ R2 \\ R3 \\ R4 \\ R5 \\ R6^C \end{pmatrix}$$

$$^{64}H_2 = \left(\begin{array}{c|c} & C_I \\ ^{32}H_2 & ^{32}H^c_2 \\ \hline 00\ldots0 & 11\ldots1 \end{array}\right)$$

Figure 4.9: Constructing H_2-matrix for proposed (16,48,7) code

Table 4.1 shows the number of XOR gates and maximum logic depth for the syndrome generator, number of forbidden 3-cycles and forbidden 4-cycles, and the number of check bits for a set of different relevant codes.

The first set codes are for packets with 8 bit header and 24 bit data. The SEC-DBED code proposed in [Bodnar 03] can correct all single-bit errors in the packet and can detect all the double adjacent errors in the 8-bit nibbles but cannot detect the double adjacent errors at the nibble boundaries. The 3FCs corresponds to these cases. This code cannot correct double adjacent errors in the header portion. It has a very large number of

4FCs. The SEC code can only correct single-bit errors. It has a large number of 3FCs and 4FCs. The $(B_2EC)_8$-$(SEC)_{24}$ code derived using the method proposed in [Namba 01] can correct all single-bit errors and additionally can correct adjacent double-bit errors in the header part. But it cannot detect all the double adjacent bit errors in the data part because it has some 3FCs. Also it is not clear how to derive a systematic code using the method described in [Namba 01]. The hardware overhead is also larger than the proposed code. The proposed code, along with the DEC and DAEC codes are the only ones which meet all the requirements for the targeted application and it does not have any 3FC or 4FC. However the check bit overhead for the DEC and the DAEC codes are larger than the proposed code and hence they are less suitable for the router memories where memory area is a major performance constraint.

For a 64 bit packet with 16-bit header a minimum of 7 check bits are required. The SEC-DBED code proposed in [Bodnar 03] was extended for 64 bit and the derived code has a large number of 4FCs. The code cannot correct the double adjacent errors in the header portion. The SEC code also has a large number of 3FCs and 4FCs. The $(B_2EC)_{16}$-$(SEC)_{48}$ code derived from [Namba 01] has some 3FCs. The proposed code is free of all 3FC and 4FCs and hence meets all the requirements for the targeted application. Figure 4.9 shows how the XOR gate overhead varies with respect to the header size given a particular packet size for the proposed code. One nice feature for the proposed code is that the overhead varies almost linearly with the header size.

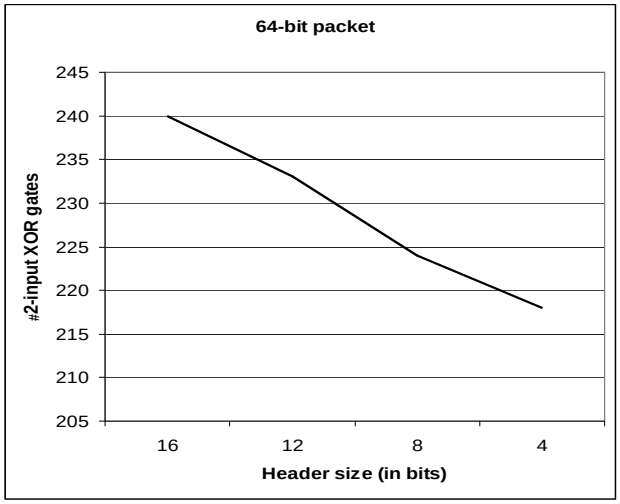

Figure 4.10: XOR gate overhead vs header size

Next we discuss the encoding and decoding strategy for the proposed code.

4.7 ENCODING/DECODING ALGORITHM

The proposed code is systematic. During encoding, the data bits can be directly copied and the check bits are generated using an XOR network corresponding to the *G*-matrix. The decoding algorithm is as follows:

1. Generate the syndrome using an XOR network corresponding to the *H*-matrix.
2. If the syndrome is the all zero vector, then no error is detected, otherwise one or more errors occurred.
3. If the syndrome matches any of the *H*-matrix columns then a single error is detected and the error position is the corresponding column position. The corresponding bit should be flipped to correct the error.

4. Else if the syndrome matches any of the *(header-size-1)* adjacent double error syndromes or the double error syndrome for the error at the boundary of the header and the data parts, then a double adjacent error is detected and the corresponding bit positions are generated using the error correction logic.

5. Else an uncorrectable error (UE) (i.e., a double non-adjacent error or more than two errors) has occurred.

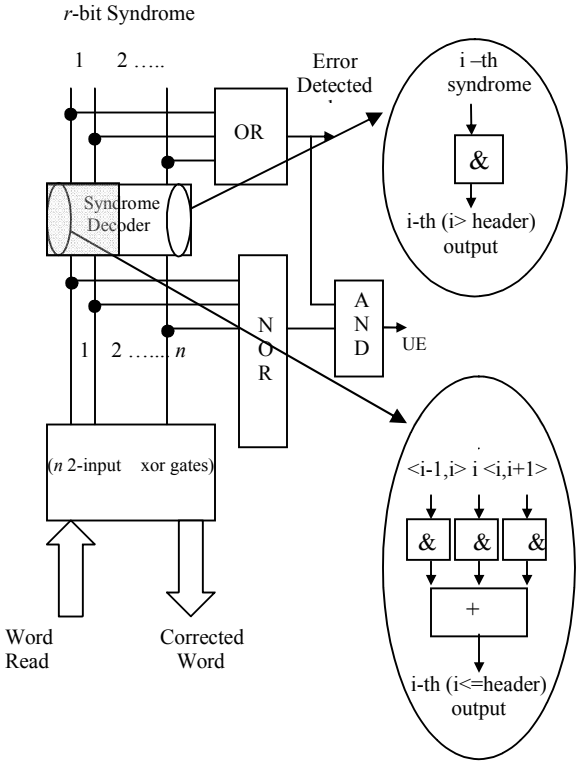

Figure 4.11: Error detection and correction block diagram

If an uncorrectable error is encountered then it is assumed that the error occurred in the data part of the packet since the header errors are always correctable. When the UE

signal is high a retransmission is requested. The header provides the address of the source and destination.

The only additional overhead with respect to a conventional SEC code comes from step 4 of the decoding step. Figure 4.10 shows the basic error detection and correction block diagram. If a non-zero syndrome is encountered, then the OR gate flags an error indication. If the syndrome matches any of the single error syndromes then the syndrome decoder generates a 1 in the erroneous bit position. Otherwise, if the syndrome matches any of the adjacent double error syndromes in the header portion, then the decoder generates 1's at the erroneous adjacent bit positions. Otherwise the output of the syndrome decoder is the all zero output. The syndrome decoder output (for the header part) consists of 3-input OR gates whose inputs are driven by outputs of *r*-input AND gates. The *i*-th output of the decoder is 1 if and only if a single error occurred at the *i*-th bit or a double-adjacent error occurred at $<i,i+1>$ bits or $<i-1,i>$ bits. For the remaining part, only an *r*-input AND gate is required to generate the *i*-th signal. The outputs of the decoder are used to generate the corrected word, by using *n* 2-input XOR gates. If the syndrome is non-zero and does not match any of the single or double-adjacent error syndromes, then an uncorrectable error (UE) is encounter and the UE signal is flagged.

4.8 CONCLUSIONS

The ECC methodology described in this chapter provides the ability to correct all single-bit errors and detect all double adjacent errors in a packet while correcting all adjacent errors in the header portion of the packet at very little cost. The proposed code has the same check bit overhead as an SEC code. The encoding/decoding overhead and latency is also similar to the SEC code. The only drawback is the possibility of

miscorrection for a small subset of multiple errors, however MBUs caused by a single SEU have a much higher probability of occurring than having multiple independent SEUs accumulating in the same word residing in the router buffer. The same holds for the link errors during transmission. The presented code provides a very low cost option to protect the packets against the most likely errors in the NoC environment by allowing different levels of protection to different parts of the packet.

Chapter 5: Iterative OPDD Based Signal Probability Calculation

There are a number of important applications where estimating signal probabilities in a circuit is necessary including determining soft error susceptibility and random pattern testability. This chapter presents an improved method to accurately estimate signal probabilities using ordered partial decision diagrams (OPDDs) [Kodavarti 93] for partial representation of the functions at the circuit lines. OPDDs which are limited to a certain maximum number of nodes are built iteratively with different variable orderings to efficiently explore different regions of the function. Signal probability bounds (upper and lower) are computed from the OPDDs. From each OPDD, information is extracted to tighten the signal probability bound and guide the variable ordering for the next OPDD. By restricting the size of each OPDD to a small number of nodes, they can be constructed and processed quickly to obtain a fast and accurate estimate of signal probabilities. Experimental results demonstrate the effectiveness of the approach compared with existing methods.

5.1 RELATED WORK

The signal probability of a net in a combinational circuit is the probability that a randomly generated input vector will produce a logic value of 1 on this net. There are a number of important applications where calculating signal probabilities in a circuit are necessary. Originally signal probability was studied in the context of pseudorandom testing to determine detection probabilities for faults. Given the detection probabilities for faults in a circuit, it is possible to compute the expected fault coverage for a particular pseudo-random pattern test length and to identify random pattern resistant faults

[McCluskey 88]. More recently, as soft errors in logic circuits have become an important issue, signal probability is also needed in this context for determining the probability of a single event upset (SEU) propagating to a latch. Knowing the soft error susceptibility of nodes in a circuit allows better insertion of soft error protection schemes and better selection of error detecting codes. For circuits that do not have reconvergent fanout, signal probabilities can be computed exactly in linear time. However, in the general case where reconvergent fanout exists, computing signal probabilities is NP-hard [Parker 75]. A wide variety of techniques have been developed for estimating signal probabilities which provide various degrees of accuracy and runtime. A fast and simple approach, used in COP [Brglez 84], is to assume all signals in the circuit are independent, however, this can lead to large inaccuracies due to correlations between signals from reconvergent fanout. COP can be improved by estimating the impact of correlations using cofactors [Al-Kalahi 97] and first order Taylor expansion [Uchino 97]. Partitioning the circuit into "supergates" which totally enclose reconvergent fanout can be used to speedup signal probability calculations [Seth 85, 89], [Chakravarty 90]. Another approach for estimating signal probabilities is to use sampling simulation [Jain 85], [Wunderlich 85], [Reijimon 05]. One class of techniques computes signal probability bounds (upper and lower) which has the nice property of not only estimating signal probability, but also bounding the maximum error in the estimate. One such technique is the "cutting algorithm" [Savir 80] which cuts fanout lines in the circuit to make it fanout-free and then assigns a probability bound of [0,1] to the cut-lines. Techniques for tightening the bounds obtained with the cutting algorithm include [Markowsky 87] which uses a blocking heuristic to reduce the number of cuts, [Savir 90] which combines it with the Parker-McCluskey method [Parker 75], and [Kapur 92] which uses conditional probabilities to tighten the initial bounds on

the cut lines. Another technique for computing signal probability bounds is to use ordered partial decision diagrams (OPDDs) as described in [Kodavarti 93]. If the full BDD [Bryant 86] was known, then exact signal probabilities could be computed by simply counting the paths that go to the terminal 1 node. However, constructing a full BDD can be exponential in the number of inputs and thus is not practical in many cases. The idea in [Kodavarti 93] is to construct an OPDD which is limited to a certain maximum number of nodes (thereby limiting both time and memory). A bound on the signal probability is then obtained from the OPDD which contains the efficiently obtainable implicants (which typically includes the largest ones). By using a few different variable orderings and saving the best upper and lower bound seen for any ordering, it was shown in [Kodavarti 93] that the signal probability could be computed quite accurately in very short time. This chapter presents a new method for using OPDDs to estimate signal probability that gives significantly tighter bounds than the method in [Kodavarti 93]. There are two key ideas in the proposed method. The first is to constructively combine information obtained in one OPDD with the next. In [Kodavarti 93], the probability bounds are computed independently for each OPDD and then the best upper bound across all OPDDs is combined with the best lower bound across all OPDDs to form the final bound. In the proposed method, the implicants from each OPDD are extracted and combined together when computing the final upper and lower bound. Double counting is avoided by making the implicants disjoint. The second key idea is to use the information from previous OPDDs to guide the selection of the variable ordering for the next OPDD. Here a heuristic algorithm for selecting the variable orderings to efficiently explore the unknown space is presented. The proposed variable ordering algorithm uses information about the implicants found so far to help in finding new implicants to tighten the bounds

further. Experimental results are shown which demonstrate the effectiveness of the proposed method. Note that there has been some work in the domain of model-checking that uses over-approximating and underapproximating of BDDs [Ravi 98]. However, this is fundamentally different than what is done here because it involves first building the full BDD and then compressing it by discarding nodes in a deterministic manner. What is done here is to avoid building a full BDD, but rather iteratively build small limited-size partial BDDs that can be processed very quickly with no risk of "blowing up."

5.2 COMBINING INFORMATION ACROSS OPDDS

OPDDs are a variant of ordered binary decision diagrams (OBDD) introduced in [Brayant 86]. Partial information is obtained by restricting the number of nodes when building the graph to a constant *k*. The missing information is represented by an UNKNOWN (U) terminal node. Figure 3.1 shows the full OBDD representing the function: $ab'+ac'+b'c$. The 0-arcs are represented with dashed lines and the 1-arcs are represented with solid lines. Figure 3.2 through 3.4 shows corresponding OPDDs for various variable orderings. An OPDD can have 3 terminal nodes namely the 0-node, 1-node, and U-node. Variable ordering plays an important role in the amount of information that can be gathered from an OPDD. OPDDs can be built with different variable orderings to explore different regions of the function. From an OPDD, the 0-probability, 1-probability, or U-probability can be computed as described in [Kodavarti 93]:

1) Initialize the sum S_r of the root node n_r to $S_r = 1$.

2) Initialize the sum S_i of all nodes n_i, for all $i \neq r$, to $S_i = 0$.

3) Associate a line probability (p_i) for each circuit input x_i, for all i.

4) At every non-terminal node n_i, representing input variable x_i, perform two operations (during breadth first traversal of the OPDD):

Add $(1-p_i)*S_i$ into the sum of the 0-arc child

Add p_i*S_i into sum of the 1-arc child

5) Calculate the lower bound of the signal probability from the final value of the sum, S_1, at the ONE terminus, and the upper bound including the sum, S_U, at the UNKNOWN terminus.

Lower bound = S_1

Upper bound = S_1+S_U

One limitation of the approach in [Kodavarti 93] is that the information is not retained across the OPDDs. The proposed algorithm maintains a global disjoint cube cover across different OPDDs to obtain tighter bounds on the signal probability. Note that for the rest of the chapter 1-path, 0-path, and U-path will denote paths terminating at 1-terminus, 0-terminus, and U-terminus, respectively. The corresponding cubes will be denoted as 1-cube, 0-cube, and U-cube, respectively. The 1-paths in the OPDD encode cubes covering some part of the ON-set of the function represented at the root node. The 0-paths do the same for the OFF-set. By construction of the OPDD these cubes are disjoint. By constructing OPDDs with different variable orderings, different regions of the function can be explored. From each OPDD, disjoint cubes are collected and a global list of such cubes is maintained both for the OFF-set (*g0-cov*) and the ON-set (*g1-cov*). The cube list is maintained as a sorted list in decreasing order of the cube sizes. The larger the number of don't cares (X's), the larger is the cube and the larger is its contribution to the signal probability of the line. While adding a new cube to the current global cube cover, three things are checked:

1) If it is contained in one of the existing cubes then it is not added. This is checked by iterating over all the bits of the cubes. If the smaller cube matches with the larger cube at all the specified bit positions then it is contained in the larger cube.

2) If it is mutually disjoint with each of the existing cubes, then it is simply added to the list and the sorted order is maintained. This can be easily checked by looking for a conflict in any of the specified bit position of the cube to be added with each of the existing cubes in the cover.

3) If the cube to be added overlaps with one or more of the existing cubes in the cover then the cube is made disjoint and added to the list. This is done the following way. If two cubes have overlap then the cubes are traversed bit by bit and for the first bit position where one of the cubes has a specified bit and the other cube is unspecified, the unspecified bit is specified with the opposite value of the specified bit value of the other cube.

At the end of all the iterations, the final lower bound on the 1-probability is computed directly from the global disjoint cube cover (*g1*-cov) by simply adding the probabilities of the individual disjoint cubes. Since they are disjoint, the probabilities are independent and can simply be summed together. The ambiguity u is computed as [$u=1-(g0+g1)$] where $g0$ and $g1$ are the probabilities computed from the *g0*-cov and *g1*-cov respectively. The final upper bound is equal to the lower bound plus the ambiguity u. Accuracy versus runtime can easily be traded off in selecting the OPDD size limit as well as the number of OPDDs that are built. The proposed approach can be used with any set of OPDDs derived with any variable ordering. However, in the next section, a heuristic approach for guiding the selection of the variable ordering to find better cubes for the global cover is described. The example in Figs. 3.2-3.4 illustrates the proposed approach

on a very small example. Figure 3.1 shows the OBDD representation of the function ab' +ac' +b'c. The complete ON-set and OFF-set of the function are shown on a Karnaugh map. Figure 3.2 shows the OPDD with variable ordering <a,b,c> where the bound on the maximum number of non-terminal nodes is 3. After the first OPDD, *g0*-cov contains {01x} and *g1*-cov contains {10x}. The bound on signal probability computed from the OPDD in Fig. 3.2 using the method in [Kodavarti 93] is [.25,.75]. The OPDD in Fig. 3.3 uses the variable ordering <b,a,c>. No new cubes are added to either *g0*-cov or *g1*-cov as the cubes found are already present in the respective sets. The third iteration uses the variable ordering <a,c,b>. After the third iteration, one new 1-cube (1x0) is found. This is made disjoint with the existing cube in *g1-cov* and is added to *g1-cov*. The contents of *g1*-cov after this iteration is {10x,110}. The probability computed from this set is (.25+.125) = .375. Similarly the contents of the g0-cov after the 3rd iteration is {01x,000}. The probability computed from g0-cov is (.25+.125) = .375. The ambiguity after 3rd iteration is computed as [1 – (.375+.375)] = .250. So finally the estimated signal probability bound after 3rd iteration would be [g1,g1+u] = [.375,.625]. The actual signal probability in this case is .5. Note that if we don' t use the global cube covers to learn across multiple OPDDs then the best bound that can be obtained after 3rd iteration under the same variable orderings is [.25,.75].

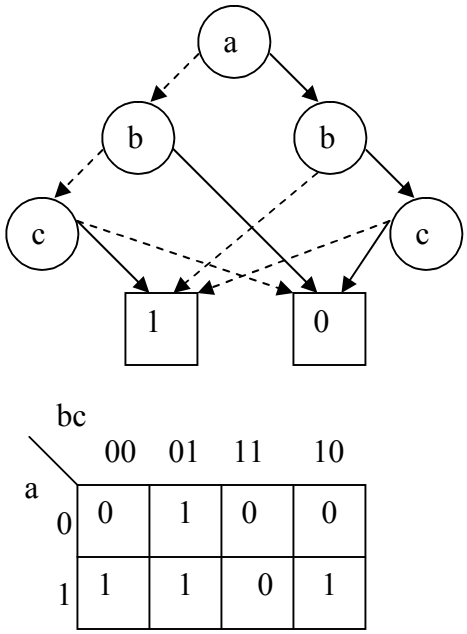

Figure 5.1: Binary decision diagram and Karnaugh map

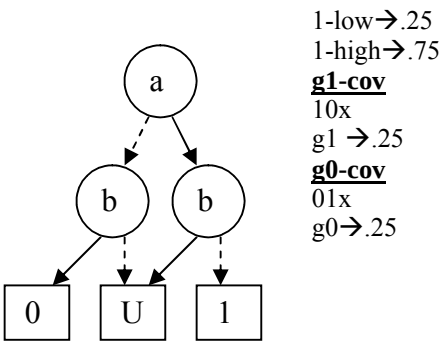

Figure 5.2: OPDD with variable ordering <a,b,c>

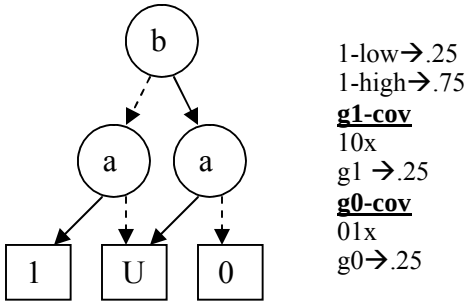

Figure 5.3: OPDD with variable ordering <b,a,c>

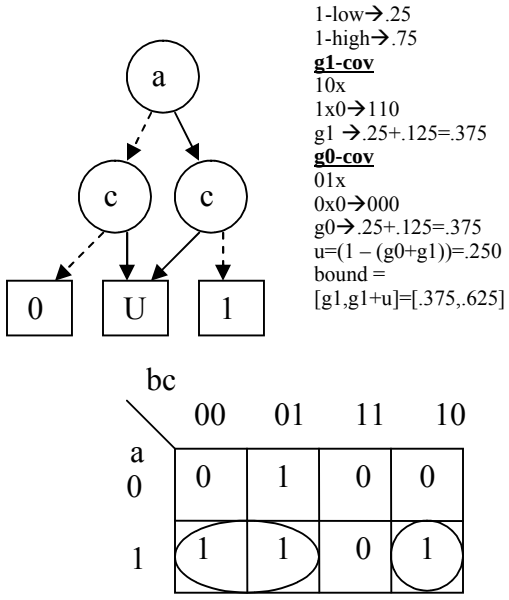

Figure 5.4: OPDD with variable ordering <a,c,b>

5.3 UNKNOWN SPACE EXPLORATION

While building an OPDD, whenever the number of nodes exceeds the predefined bound on the number of the nodes, the unknown U terminus is created and all paths are directed to it. The number of paths and the length of paths ending at the U terminus determine the size of the unknown space and hence the ambiguity in the signal probability. The variable ordering used when building an OPDD has a big impact on the resulting composition of the unknown space. The proposed method involves iteratively building multiple OPDDs and extracting collective information, and thus the goal in selecting the variable ordering for each additional OPDD is to try to explore the unknown space from the previous set of OPDDs. One common approach for variable ordering is to perform a depth first search (DFS) of the circuit from primary outputs (POs) to primary inputs (PIs) and append a PI to the ordering as soon as it is traversed. When performing a DFS, a decision has to be made at each gate as to in what order its inputs should be traversed. A number of different heuristics have been developed for making this decision (e.g., [Malik 88] and [Fujita 93]). The conventional heuristics for guiding the DFS are targeting the problem of minimizing the overall size of a full BDD. These conventional heuristics are very useful in forming the first couple OPDDs as they are likely to result in identifying the largest implicants. However, after information has been extracted from the first couple OPDDs using the proposed methodology, the usefulness of the conventional heuristics for variable ordering when building subsequent OPDDs diminishes because they are more likely to explore the space of the function that has already been explored in the previous OPDDs. Here we propose a new heuristic to guide the DFS so that it will lead to a variable ordering that more effectively explores the unknown space. The idea behind the proposed heuristic to try to measure how much of the unknown space has been

explored with respect to each variable across all the OPDDs built so far, and then direct the DFS towards the variables for which the unknown space has been least explored. The *U-effect* of a variable is defined as the sum of all the U-paths in which the variable appears in either complemented or uncomplemented form weighted by the size of the corresponding U-cube for the U-path. Thus, the U-effect of a variable is computed as (2^{n-ki}) for each U-path i in which the variable appears, where ki is the number of nodes along the path and n is the number of primary inputs of the circuit. The U-effect of the variables for the OPDD in Fig. 5.2 is shown in Table 5.1. There are two U-paths each of which include variables *a* and *b.* The size of the U-cube corresponding to each path is 2, and thus the U-effect for *a* and *b* is 4. Since variable *c* does not appear on any U-paths, its U-effect is 0.

Table 5.1: U-effect for OPDD in Fig 5.2

Variable	Effect
a	2+2=4
b	2+2=4
c	0

Table 5.2: Cumulative U-effects of OPDDs in Figs 5.2-5.4

Variable	Effect
a	12
b	8
c	4

As each new OPDD is constructed, the U-effect for each variable is computed and added to the U-effect of the previous OPDDs. Thus, a running total of the Ueffect over all the OPDDs is maintained for each variable. The running total of the U-effect over the three OPDDs in Figs. 5.2-5.4 is shown in Table 5.2. The cumulative U-effect for a set of

OPDDs gives a rough measure of how well the unknown space has been explored with respect to each variable. This is then used as a heuristic to direct the DFS towards the variables that have the lowest U-effect. This helps to explore the least explored region of the function and hopefully construct an OPDD in which new implicants can be obtained to further reduce ambiguity in the signal probability calculations. This is illustrated in the small example in Fig. 5.5 where using the U-effects from Table 5.2 result in a new ordering <c,b,a> for which the OPDD built as shown in Fig. 5.5. For the OPDD in Fig. 5.5, the U-space vanishes and the exact probabilities can be obtained. When searching the unknown space using the U-effect heuristic, the DFS is modified so that the decision as to which order in which to traverse the inputs of a gate are made based on the support set for each input. The support set for an input is the set of PIs (variables) that it depends on. The U-effect of each variable in the support set is summed together, and the inputs of the gate are traversed in reverse order of their total U-effect. So the overall strategy for iteratively building the OPDDs is the following. Conventional heuristics for variable ordering are used to build the first couple OPDDs to identify large implicants, and then the proposed heuristic of using the U-effect is used when building subsequent OPDDs to more effectively search the unknown space.

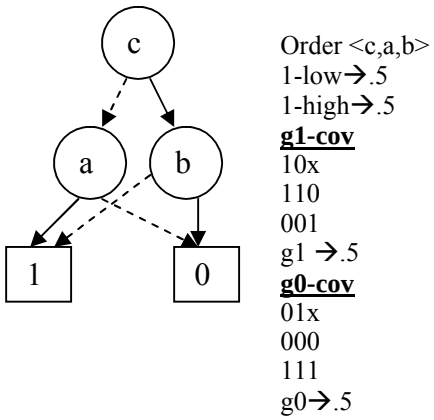

Figure 5.5: OPDD using USSE generated ordering

5.4 RUNTIME COMPLEXITY ANALYSIS

In this section, the runtime complexity for the proposed method is analyzed and compared with [Kodavarti 93]. The complexity is described below in terms of the following: number of gates in the circuit (*G*), number of primary inputs in the circuit (*N*), limit on the maximum number of nodes in the OPDD (*B*), and number of iterations used (*I*) where one OPDD is built in each iteration. Note that *B* and *I* are actually constants that do not scale with circuit size. They are shown in the equations below just for completeness and to aid the reader's understanding.

Compute Variable Ordering – Initially this is done with conventional heuristics which is the same as for [Kodavarti 93]. The complexity depends on which heuristics are used, but good results for DFS based methods can be obtained in linear time in the number of gates. Thus the overall complexity is *O(GI)* since it needs to be done for each OPDD iteration. In the proposed method, after the first couple OPDDs, then the U-effect heuristic described in Sec. 5.3 is used. The U-effect for each variable can be computed by

traversing each OPDD which is *O(BI)* and then it needs to be sorted which is *O(Nlog(N)I)*. The support set for each line in the circuit can be obtained in *O(G)* and needs to be computed only once and stored. So the overall complexity for DFS with the U-effect heuristic is *O(BI+Nlog(N)I +G)*.

- *Build OPDDs* – The complexity for building the OPDDs is $O(GB^2I)$ which is the same as for [Kodavarti93].
- *Construct Global Disjoint Cover* – This is unique to the proposed method. Note that the number of cubes cannot be larger than the number of nodes in the OPDDs. Hence this can be done in $O(NGB^2I^2)$. Note that these are very fast bitwise comparison operations.
- *Calculate Signal Probability Range* – This is just a matter of adding the size of all the disjoint cubes. For both [Kodavarti 93] and the proposed method, this is *O(GBI)*. As mentioned before, *B* which is the node limit for the OPDDs and *I* which is the number of iterations are constants that do not scale with circuit size. So if we only consider the factors that scale with circuit size, then the overall complexity for the proposed method is *O(NG)* compared with *O(G)* for [Kodavarti 93]. For realistic industrial circuits, the extra *N* factor is not very significant for two reasons. One is that while in the worst case the number of inputs in the largest cone of logic could theoretically be equal to *N*, typically it is relatively small and doesn't scale up much for larger designs (it depends mostly on the function being implemented and not on the integration density), and the other reason is that the extra complexity is coming from the bitwise comparison operations for the cubes which can be done very quickly.

Consequently, the actual runtimes for the two methods is very similar and on the order of a minute for the ISCAS circuits which have thousands of gates. If we scale up to millions of gates (1000 times larger than the ISCAS circuits), the runtime would still be less than one day in the worst case assuming only a single processor was used. In practice, a hierarchical design could be easily partitioned and spread over multiple processors, not to mention there is likely to be economies of scale.

5.5 EXPERIMENTAL RESULTS

The proposed approach is specifically useful for large circuits where building the full BDD is not possible. However, for comparison with other methods, experiments were performed on the ISCAS 85 benchmark circuits even though it is practical to build a full BDD for most of those circuits with 4000 nodes. In [Kodavarti 93], results were reported for using a limit of 500 nodes in the OPDDs with 4 iterations (i.e., building 4 OPDDs with different variable orderings). These results from [Kodavarti 93] are shown in Table 5.4 along with the results reported in [Kodavarti 93] for the cutting algorithm which are shown in Table 5.3. Experiments were performed using the proposed method with the same parameters, namely a 500 node limit for each OPDD and 4 iterations. The results for the proposed method are shown in Table 5.4. In each of these tables, the number of lines with different ranges of ambiguity between 0% and 100% are shown. As can be seen from the results, the proposed method is able to reduce the amount of ambiguity in the signal probability ranges considerably in all the circuits compared with [Kodavarti 93]. Further experiments were performed to see how the results varied with the number of iterations. Fig. 5.6 shows results for *C880* where the total ambiguity

normalized with respect to the first iteration is shown on the y-axis and the number of iterations is shown on the x-axis. A 200 node limit was used for the OPDDs. As the number of iterations increases, the ambiguity decreases. There tends to be a diminishing marginal return, however, and the amount of improvement from one iteration to the next can vary due to the fact that heuristics are used.

Table 5.3: Cutting Algorithm

Ckt Name	Ambiguity ranges (in %)							
	#line	0	>0 <=30	>30<=50	>50<=80	>80<=90	>90<100	100
c432	160	27	27	27	0	0	6	73
c499	202	13	0	7	20	5	18	139
c880	383	19	9	45	23	0	0	287
c1355	546	9	1	30	8	0	0	498
c1908	880	52	0	37	4	0	0	787
c2670	1269	154	6	136	38	5	55	878
c3540	1669	53	0	53	14	1	0	1548
c5315	2307	137	5	154	28	0	0	1893
c6288	2416	3	0	58	1	0	0	2354
c7552	3513	120	5	190	91	0	0	3107

Table 5.4: [Kodavarti 93]

Ckt Name	Ambiguity ranges (in %) : Max node limit = 500							
	#line	0	>0 <=30	>30<=50	>50<=80	>80<=90	>90<100	100
c432	160	94	31	14	16	5	0	0
c499	202	137	33	32	0	0	0	0
c880	383	354	29	0	0	0	0	0
c1355	546	338	143	64	0	1	0	0
c1908	880	731	149	0	0	0	0	0
c2670	1269	1237	24	2	6	0	0	0
c3540	1669	1261	119	80	179	28	2	0
c5315	2307	2227	66	6	6	0	0	0
c6288	2416	1281	394	36	231	100	310	64
c7552	3513	3387	112	8	6	0	0	0

Table 5.5: Proposed

Ckt Name	Ambiguity ranges (in %) : Max node limit = 500							
	#line	0	>0 <=30	>30<=50	>50<=80	>80<=90	>90<100	100
c432	160	158	2	0	0	0	0	0
c499	202	193	7	2	0	0	0	0
c880	383	360	23	0	0	0	0	0
c1355	546	436	102	8	0	0	0	0
c1908	880	816	64	0	0	0	0	0
c2670	1269	1240	21	8	0	0	0	0
c3540	1669	1484	87	98	0	0	0	0
c5315	2307	2307	0	0	0	0	0	0
c6288	2416	1292	383	68	222	114	272	63
c7552	3513	3408	99	2	4	0	0	0

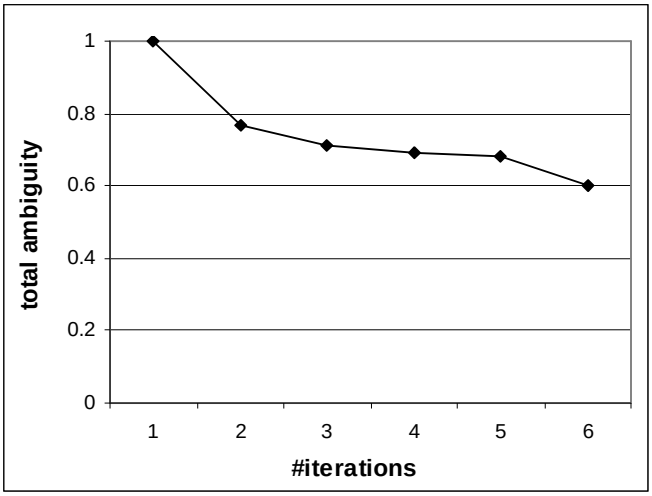

Figure 5.6: #iterations vs total ambiguity (normalized) (c880) [max #nodes = 200]

5.6 CONCLUSIONS

In this chapter we proposed an iterative technique to compute signal probabilities using disjoint cube cover obtained from OPDDs. We also proposed a variable ordering algorithm to reduce ambiguity in the computed signal probabilities by efficiently exploring the unknown solution space obtained from explored OPDDs.

Chapter 6: Using Limited Depth Sequential Expansion for Decompressing Test Vectors

6.1 INTRODUCTION

Test vector compression involves storing a deterministic test set on the tester in a compressed form and using on-chip hardware to decompress it. The test data bandwidth between the tester and chip is generally a bottleneck, so compressing the amount of data that needs to be transferred reduces test time. Test vectors are highly compressible because typically only 1-5% of the bits are specified (care) bits while the rest are don't cares. A number of commercial tools have been introduced in recent years for implementing test vector compression.

One class of test vector compression schemes that is used in a number of commercial tools is based on using a linear decompressor to expand the compressed data coming from the tester to fill the scan chains. Any decompressor that consists of only XORs and flip-flops is a linear decompressor and has the property that its output space (i.e., the space of all possible vectors that it can generate) is a linear subspace. Determining whether a particular test cube (i.e., test vector in which the unassigned inputs are left as don't cares) can be encoded by a linear decompressor can be done by solving a system of linear equations for the specified (care) bits. Combinational linear decompressors [Bayraktaroglu 03], [Mitra 06], use only XOR networks with no flip-flops. If there are b tester channels expanding to fill n scan chains (as illustrated in Fig. 6.1), then each n-bit "scan slice" is encoded with the b free-variables coming from the tester in the corresponding clock cycle (each bit stored on the tester can be considered a "free-variable" that can be assigned 0 or 1). One drawback is that the worst-case most

highly specified scan slices tend to limit the amount of compression that can be achieved because the number of channels from the tester needs to be sufficiently large to encode them. Sequential linear decompressors [Krishna 01, 04], [Konemann 01], [Rajski 04] use linear finite state machines such as linear feedback shift registers (LFSRs) or ring generators [Mrugalski 04] which retain free-variables received from the tester in earlier clock cycles thereby allowing a scan slice to be encoded using free-variables across multiple clock cycles. This allows greater flexibility to handle heavily specified scan slices that may occasionally occur. Consequently, for a fixed number of tester channels, sequential linear decompressors have a higher probability of being able to encode a given test cube.

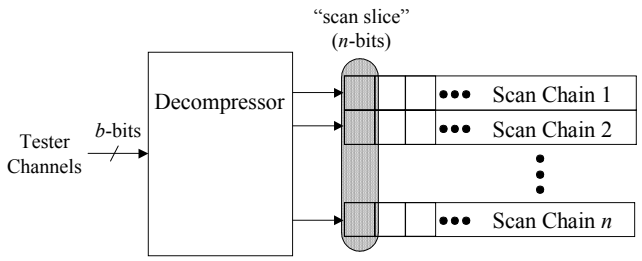

Figure 6.1: Block diagram of test vector decompression

Another class of test vector compression schemes that is used in commercial tools is based on broadcasting the same value to multiple scan chains. This concept was first proposed in [Lee 98] for scan chains driving independent circuits, and then was adapted for scan chains driving dependent circuits in [Hamzaoglu 99] by adding a serial mode for applying test cubes that cannot be applied in broadcast mode (this has come to be known as Illinois scan). Illinois scan is essentially a special degenerate case of linear decompression in which the decompressor consists of only fanout wires (no XOR gates).

The encoding flexibility for the broadcast mode of Illinois scan is less than linear decompressors that use XOR gates. Given a particular test cube, the probability of encoding it with a linear decompressor that uses XORs is higher than with Illinois scan because it has a more diverse output space with fewer linear dependencies than a fanout network does. However, the fact that faults can be detected by many different test cubes provides an additional degree of freedom. The advantage of Illinois scan is that it is very easy to incorporate the constraints imposed by the decompressor during the ATPG to exploit this degree of freedom in choosing a test cube. This can be done by simply tying dependent inputs together in the circuit description given to the ATPG so that the ATPG algorithm will search only for encodable test cubes. For linear decompressors that use XORs, the conventional approach is to first generate a test cube and then solve the linear equations to see if it is encodable and if it is not then try to find a different test cube with possibly less aggressive dynamic compaction. This is a two step process that does not incorporate the constraints in the ATPG search/backtrace procedure as is done with Illinois scan. So each approach has its advantages. Linear decompressors that use XORs can encode a wider range of test cubes than Illinois scan, but Illinois scan can harness the ATPG to search for encodable test cubes more efficiently.

Ideally, it would be nice to combine the advantages of both approaches. In other words, have greater encoding flexibility than Illinois scan, but retain the ability to include the constraints in the ATPG search/backtrace so the ATPG can efficiently find encodable test cubes. Some work has been done in this direction. One approach is to provide the ability to reconfigure the broadcast mode in Illinois scan to change the constraints. This can be done statically (on a per scan basis) by either reconfiguring the scan chains [Pandey 02] or reconfiguring the fanout network [Samaranayake 03], [Tang 03], [Mitra

06]. Or, it can be done dynamically (on a per shift basis) [Sitchinava 04] where MUXes are placed in front of each scan chain and the control signals for the MUXes are driven by tester channels. In [Wang 04], a combinational network that includes XOR gates is also used and included in the ATPG backtrace.

All of the previous schemes that include the decompressor constraints in the ATPG backtrace are based on combinational decompression where each scan slice must be encoded using only the b free-variables arriving from the tester in a single clock cycle. In this chapter, we investigate how to use sequential decompression in a way that the constraints are included in the ATPG search/backtrace. The advantage of sequential decompression is that free-variables across multiple clock cycles can be used to encode each scan slice thereby providing greater flexibility and alleviating the problem of the worst-case most heavily specified scan slice limiting the encodability of a test cube. Conventional sequential linear decompressors based on LFSRs or ring generators are not amenable to including the constraints in the ATPG backtrace. The reason is that typically the value of each scan cell depends on the XOR of a large number of free-variables. Including these types of constraints in the ATPG backtrace would greatly increase the search complexity for the ATPG resulting in a large number of backtracks and aborts thereby rendering the ATPG ineffective. To get around this problem, this chapter proposes the use of limited dependence sequential expansion which keeps the constraints to a minimum to allow effective ATPG backtrace while still retaining the advantages of sequential expansion in terms of using free-variables across multiple clock cycles.

The contributions of this chapter include the following:

- A systematic study of different ways of increasing the flexibility of decompressors for a fixed number of tester channels.

- A new decompressor design that uses limited dependence sequential expansion is proposed, and a synthesis procedure is presented.
- The probability of encoding a test cube with different decompressor designs is analyzed, and the advantages of sequential decompression are quantified.
- Experimental results for benchmarks are shown comparing different compression schemes in terms of the ATPG runtime and the amount of compression achieved.

The chapter is organized as follows: Sec. 6.2 analyzes the encoding flexibility provided by different combinational decompressor designs. Sec. 6.3 investigates the use of sequential decompressors and shows the advantages compared with combinational decompressors. Sec. 6.4 discusses some of the issues involved in selecting a decompressor design. Sec. 6.5 presents a synthesis procedure for synthesizing limited dependence sequential linear decompressors. Sec. 6.6 shows the experimental results. Sec. 6.7 is a conclusion.

6.2 COMBINATIONAL ENCODING FLEXIBILITY

Illinois scan, where a fanout network from the tester channels is used, provides the simplest constraints for ATPG since it involves simply tying inputs together. However, it has limited encoding flexibility because if two specified bits in a scan slice have opposite value and are fed by the same tester channel, they cannot be encoded. If the number of tester channels is *c*, and the expansion ratio (i.e., the ratio of scan chains to tester channels) is *k*, then the probability of not being able to encode two specified bits in a scan slice is:

$$\frac{k-1}{2(ck-1)}$$

Increasing the encoding flexibility requires adding some gates to the decompressor. Consider adding one 2-input gate to drive each scan chain. To maximize the output space of the decompressor (and hence its encoding flexibility), the logic driving each scan chain should have an output space that is equally balanced between 0's and 1's. This rules out using an AND or OR gate. The only 2-input gate whose output space is equally balanced is a 2-input XOR/XNOR gate. Note that the presence or absence of inversion does not change the probability of encoding an arbitrary test cube, so without loss of generality, only XOR will be considered. If each scan chain is driven by a 2-input XOR of a unique combination of tester channels, then if there are exactly $ck = C_2^c$ scan chains, all scan slices with 2 specified bits can be encoded and the probability of not being able to encode 3 specified bits will be less than:

$$\frac{1}{2(ck-1)}$$

So it is more likely to be able to encode 3 specified bits with 2-input XOR gates than it is to encode 2 specified bits with Illinois scan which is a considerable improvement. The cost is that the constraints during ATPG now require adding 2-input XOR gates into the ATPG backtrace for each pseudo-primary input (pseudo-PI) corresponding to a scan cell. This is illustrated in Fig. 6.2 with a small example where 2-input XOR gates are used to drive each scan chain, and the constraints for the decompressor are expanded into the circuit given to the ATPG (note that A_i, B_i, and C_i are the free-variables arriving from the tester during clock cycle i). To justify a 0 on a pseudo-PI, there are now 2 different ways to do it (assign 00 to the inputs of the XOR gate driving it or assign 11). This increases the search space for the ATPG thereby slowing it down a little. However, in comparison to Illinois scan, the ATPG has more flexibility when targeting a fault which can lead to better dynamic compaction and less need for resorting to serial mode to detect faults.

To achieve even greater flexibility, 3-input gates could be used to drive each scan chain. In this case there are two options for a balanced output space, a 3-input XOR or a 2-to-1 MUX (other balanced functions are equivalent to those two with inversion). In [Mitra 06], it was shown that using 3-input XORs can guarantee that any 3 specified bits in a scan slice can be encoded. For a MUX, one approach would be to partition the tester channels into control and data where the control channels drive the select signal for the MUXs and the data channels drive the data inputs to the MUXs. This is effectively what is done in [Sitchinava 04]. The other option would be to simply connect any combination of 3 tester channels to each MUX with no distinction between control and data.

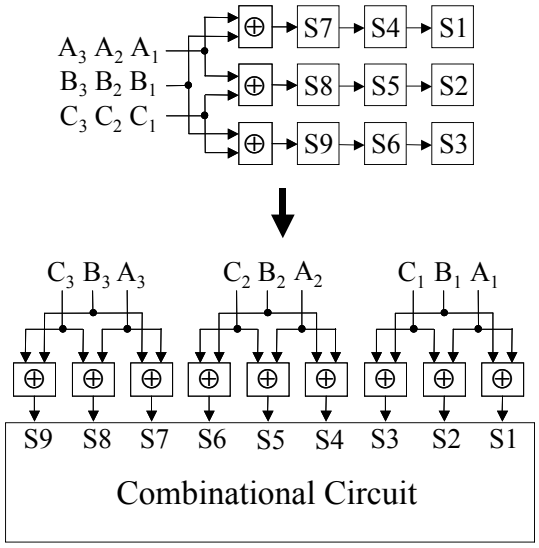

Figure 6.2: Example of including decompressor constraints at pseudo-PI's for ATPG

The graph in Fig. 6.3 shows the probability of encoding different numbers of specified bits in a single scan slice for different decompression networks when expanding from 16 tester channels to 160 scan chains (i.e., an expansion ratio of 10). The *x*-axis is the number of specified bits in the scan slice, and the *y*-axis is the percentage of all possible combinations of that number of specified bits that can be encoded. As can be seen from the graph, all the decompression networks can always encode 1 specified bit. However, as the number of specified bits is increased, the probability of being able to encode the scan slice drops. Since Illinois has the least encoding flexibility, it has the lowest probability of being able to encode a scan slice. The results for using MUXs are shown for two cases. One is where the control and data lines are separated, i.e., one of the tester channels is dedicated to driving the select line and the other 15 tester channels are used to drive the data lines. The other is where combinations of all 16 tester channels are used to drive either the select or data lines of the MUXs. The results indicate that greater

encoding flexibility can be obtained by not having a separate control line. Another interesting result is that using 2-input XOR gates is not as good as using MUXs for low numbers of specified bits, but becomes better than MUXs when the number of specified bits is equal to 10 or more. Using 3-input XORs provides considerably better encoding flexibility although it comes with the tradeoff of adding more complexity to the ATPG than the others.

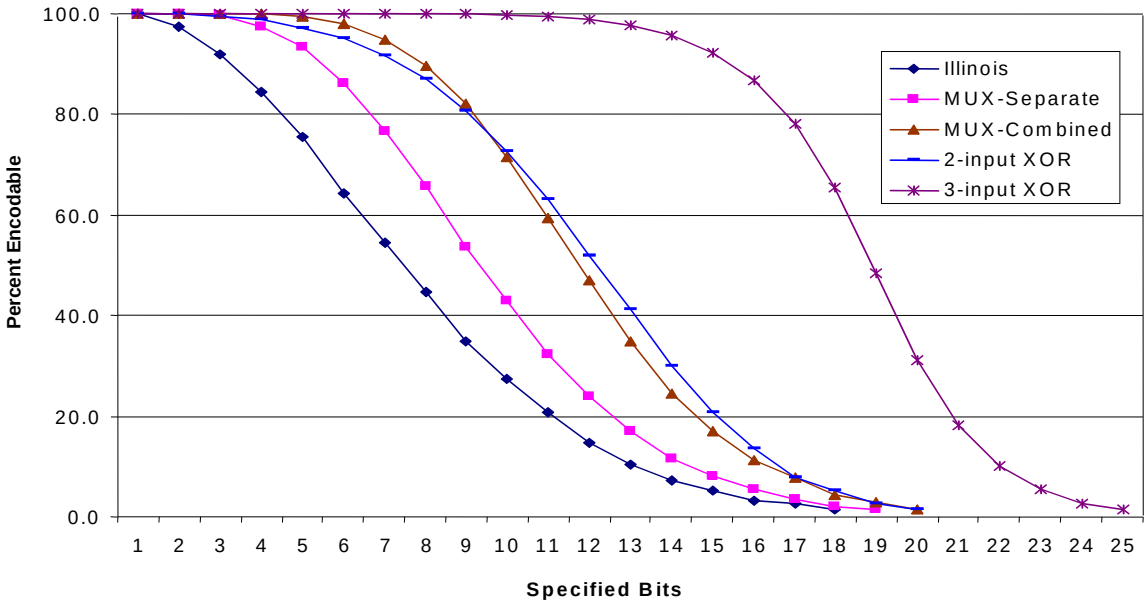

Figure 6.3: Probability of encoding scan slice for 16 tester channels expanding to fill 160 scan chains

6.3 SEQUENTIAL ENCODING FLEXIBILITY

Previously proposed schemes that include the decompressor constraints in the ATPG backtrace are limited to combinational decompression where each scan slice is encoded using only the free-variables arriving from the tester in a single clock cycle. To achieve greater flexibility to handle the heavily specified scan slices, the use of sequential decompression is investigated here since it allows free-variables across multiple clock cycles to be used in encoding each scan slice. The constraints for conventional approaches for sequential linear decompression that use LFSRs or ring generators are very complex because each scan cell can depend on the XOR of a large number of free-variables. Incorporating such complex constraints in the ATPG backtrace can greatly increase the search complexity of the ATPG. Consider a pseudo-PI whose value depends on the XOR of q free-variables. In order to justify a logic value at the pseudo-PI, q inputs need to be assigned, and the number of possible ways to assign them to get either a 0 or 1 would be 2^q-1. As q increases, this search space grows exponentially. For this reason, conventional approaches that use LFSRs or ring generators do not attempt to include the constraints in the ATPG backtrace. Instead they do ATPG and then check the constraints afterwards. The drawback of this approach is that if an encodable test cube for a fault exists, there is no guarantee that it will be found, and dynamic compaction may need to be done less aggressively in order for the linear equations to remain solvable.

In order to efficiently include the decompressor constraints in the ATPG backtrace, the constraints need to be limited to a dependence on only 2 or 3 free-variables to keep the search space manageable. If the b bits coming from the tester in each clock cycle are defined as a "tester slice", then one way to perform sequential decompression is to store the last one or two tester slices in a register and use either 2 or 3 input XOR gates

to drive each scan chain. The inputs to these XOR gates can come from the domain of the current tester slice and any previous tester slice stored in a register (this is illustrated for two registers in Fig. 6.4). This provides two benefits. One is that free-variables across 2 or 3 tester slices are used to encode each scan slice which gives more flexibility by providing access to a larger pool of free-variables to handle an occasional heavily specified bit slice, and the second benefit is that a larger number of unique free-variable combinations can be used to drive the scan chains each clock cycle. For example, if there are only 8 tester channels and 2-input XORs are used, then for a combinational decompressor there are only C_2^8 = 28 unique free-variable combinations in each clock cycle which must be broadcast to multiple scan chains if there are more than 28 scan chains. However, if one register is used to store the previous tester slice, then there are C_2^{16} = 120 unique free-variable combinations in each clock cycle, or if two registers are used there are C_2^{24} = 276 unique free-variable combinations. This allows more scan chains to be driven with unique combinations of free-variables which provides greater diversity in the output space and gives more encoding flexibility.

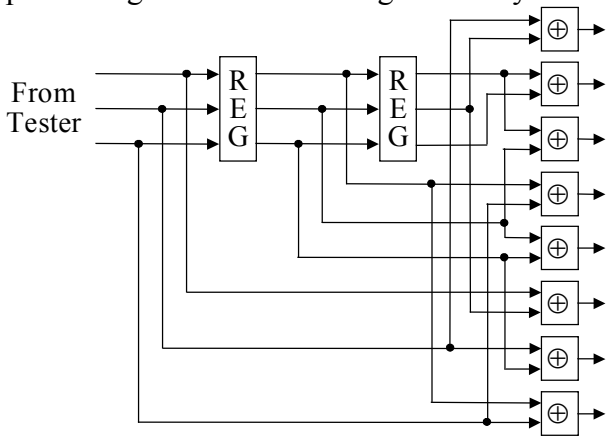

Figure 6.4: Example of limited dependence sequential decompressor with two registers

The benefit of using sequential decompression versus combinational decompression is shown in Fig. 6.5. A scan architecture consisting of 80 scan chains each 100 bits long was driven using 8 tester channels. The probability of encoding test cubes with different percentages of specified bits using different decompressor designs is shown. The *x*-axis is the percent of the bits in the test cube that are specified, and the *y*-axis is the probability of encoding the test cube expressed as a percentage. As expected, Illinois scan has the lowest probability of encoding and using an LFSR has the highest probability of encoding (a 64-bit LFSR was used with dynamic reseeding). Using 2-input XORs is shown for the case where only combinational expansion is used and then when 1 and 2 registers are used. As can be seen, the probability of encoding a test cube goes up considerably by adding the registers to perform sequential decompression. Using 2-input XORs with one register is better than using a combinational decoder with 3-input XORs. This is an interesting result because the ATPG search complexity is less with 2-input XORs than with 3-input XORs. Another significant result is the very large improvement that is achieved for 3-input XORs when one or two registers are used to store the previous tester slice. The results begin to approach what an LFSR can achieve, but in this case each pseudo-PI depends on only 3 free-variables thereby making it practical to include the constraints in the ATPG backtrace. To make the comparisons in Fig. 6.5 fair, the same number of free-variables were used for each decompressor (i.e., a total of 100 tester slices were used for encoding each test cube). No extra shifts were used to pre-load the sequential decompressors. Instead, the sequential decompressors were bypassed in the first clock cycle for the designs with 1 register and the first two clock cycles for the designs with 2 registers. The LFSR was bypassed for the first clock cycle. If one or two extra shifts are used to pre-load the sequential decompressors, the results are slightly better, but there is not much difference.

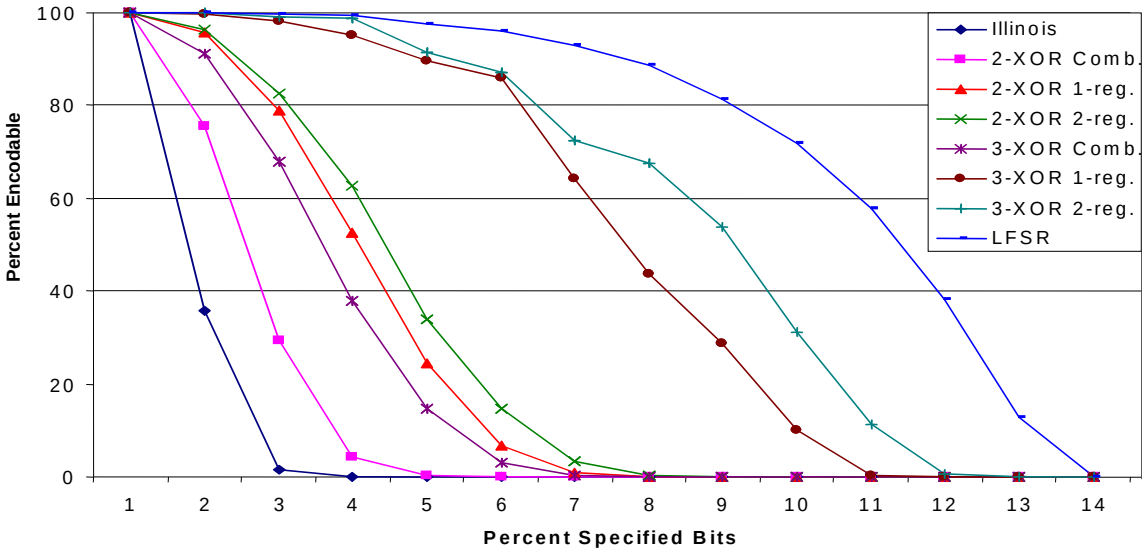

Figure 6.5: Probability of encoding test cubes for 8 tester channels expanding to fill 80 scan chains

6.4 SELECTING DECOMPRESSOR DESIGN

As seen in Fig. 6.5, the addition of registers to store previous tester slices significantly improves the encoding flexibility of the decompressor. This section discusses some of the issues involved in selecting which decompressor design to use. The first issue is whether to use Illinois scan, 2-input, or 3-input XORs. There is a tradeoff in terms of the area and ATPG time versus the amount of compression achieved. The area for all of the decompressor designs is fairly small and probably not a significant factor. The ATPG runtime is a one time cost. If some additional ATPG runtime can be handled, then the reduction in test time that can be achieved with greater compression may be very worthwhile as that reduces test costs for every chip manufactured.

The results in Fig. 6.5 show that adding registers to enable sequential decompression gives a significant boost which comes with little additional cost in ATPG runtime. The ATPG runtime will mainly depend only on whether 2 or 3 input XORs are used. Adding more registers provides a diminishing marginal improvement. Adding the first register give a big improvement, and then adding the second register give much less improvement. Using more than 2 registers will give some minor improvement, but probably not worth the cost at that point.

Another issue is how to handle the first r scan slices if r registers are used. The registers are reset between each test cube to decouple them so that each test cube is encoded with its own independent set of free-variables. This means that in the first r clock cycles for each test cube, some or all of the r registers will not yet be filled with free-variables and thus will not be ready to drive the scan chains. The simple solution is to just use r extra shifts when decompressing each test cube. The extra shifts fill the r registers before the first scan slice is decompressed. If r is very small relative to scan

length, then this will not have much impact on the test time. The other alternative is to use MUXes to bypass the sequential decompressor when decompressing the first r scan slices. The scan chains can be driven during those clock cycles with a combinational decompression network that depends only on the current tester slice. This second approach does not require any extra shifts, and this is what was used for generating all the results in this chapter to provide a fair comparison with combinational decompressors (because in this case the same number of free-variables are used to encode each test cube). However, from an implementation standpoint, the first approach of using extra shifts is probably more attractive since it simplifies the hardware.

6.5 SYNTHESIS PROCEDURE FOR DECOMPRESSOR

The procedure for synthesizing a limited dependence sequential linear decompressor for expanding b tester channels to fill n scan chains using r tester slice registers and q-input gates driving each scan chain is as follows:

Generate all $C_q^{b(r+1)}$ combinations of the current tester slice bits and the tester slice registers' bits.

For each scan chain, select an unused combination whose individual components have collectively been used as inputs to the fewest gates. Mark that combination as used. Create a gate to drive the scan chain using the selected combination as the inputs.

If there are more scan chains than combinations, then fan out the output of each gate corresponding to a combination to multiple scan chains. Keep the number of fanouts for each gate as balanced as possible.

The domain of possible inputs to the gates is the b-bits in the current tester slice combined with the b-bits in each of the registers storing the previous r tester slices. Combinations of these are selected to drive each scan chain in a way that balances the use of each input evenly. This spreads the use of the free-variables evenly.

The design could be optimized if it is customized for a particular circuit-under-test. If structural information is known about the circuit-under-test and the scan chain ordering is known, then it is possible to choose the combinations of inputs that drive each scan chain in a way that maximizes the probability of encoding a test cube. The synthesis procedure here assumes no information about the circuit-under-test, and thus generates a decompressor that is applicable for any circuit-under-test.

6.6 EXPERIMENTAL RESULTS

In this section, experimental results are presented for using different decompessors. Table 6.1 shows details for the circuits that were used. Experiments were performed on one ISCAS benchmark circuit (s38584) and two industrial circuits (Design A and B). The number of scan cells, the total number of faults, and the number of ATPG vectors required for 100% coverage of detectable faults are reported in Table 6.1.

Tables 6.2 through 6.4 report the results obtained for each of the designs listed in Table 6.1. In each case, different decompressors were used to expand 8 tester channels to fill the number of scan chains shown in the column header of the fourth, fifth, and sixth columns. The decompressors are Illinois scan, combinational 2-input XOR gates driving each scan chain, combinational 3-input XOR gates driving each scan chain, and the proposed limited dependence sequential linear decompressors with one or two tester slice registers using either 2-input XOR gates driving each scan chain or 3-input XOR gates driving each scan chain. In generating the results, the constraints for each decompressor were added to the circuit description given the ATPG tool. A commercial ATPG tool was used to generate all the results reported here (although any ATPG tool can be used).

In each table, results are first shown in the upper part of the table for using a single configuration. The results include the compression ratio that is achieved (i.e.,

normal uncompressed tester storage for a test set

generated with no constraints divided by compressed tester storage), the number of parallel and serial vectors that are used, and the coverage that is obtained if only parallel vectors are used. In the lower part of each table, results are shown for using 4 configurations with static reconfiguration where the configuration is changed only on a per scan basis. These results were obtained by first detecting as many faults as possible with the first configurations, and then using each subsequent configuration to detect any faults that still remain undetected. The results that are reported for using 4 configurations include the amount of compression, and the number of parallel and serial vectors that are required. At the bottom of the table, the ATPG runtime is shown. This is the time that it takes to run ATPG for the first configuration (subsequent configurations are much faster since most of the faults are already detected).

In the results, it can be seen that Illinois scan has the shortest ATPG runtime as expected, but it also provides the lowest amount of compression. The proposed limited dependence sequential decompressors require longer ATPG runtimes, but achieve much better compression. As can be seen, the addition of the tester slice registers to perform sequential decompression significantly improves the results.

Table 6.1: Design Details

Design	Scan cells	Faults	Fullscan ATPG vectors
s38584	1464	105298	135
Design A	7654	239902	796
Design B	856	53689	154

Table 6.2: Results for s38584

	Num. Scan Chains			
	Decompressor	192	224	256
Compression (1 config.)	Illinois	3.4	3.2	2.0
	2-xor comb	5.2	4.4	3.3
	3-xor comb	5.6	5.5	5.1
	2-xor, 2 reg	6.0	6.3	5.3
	3-xor, 2 reg	6.1	6.7	6.2
Parallel Vectors (1 config.)	Illinois	286	280	293
	2-xor comb	332	330	385
	3-xor comb	437	454	499
	2-xor, 2 reg	339	425	442
	3-xor, 2 reg	441	447	452
Serial Vectors (1 config.)	Illinois	27	31	56
	2-xor comb	11	18	28
	3-xor comb	5	7	10
	2-xor, 2 reg	5	5	11
	3-xor, 2 reg	3	3	7
Coverage with parallel vectors only (1 config.)	Illinois	93.2	94.7	93.2
	2-xor comb	99.6	98.2	93.1
	3-xor comb	99.7	98.1	93.2
	2-xor, 2 reg	99.8	98.0	93.1
	3-xor, 2 reg	99.8	99.6	95.6
Compression (4 config.)	Illinois	3.9	3.7	2.4
	2-xor comb	6.5	4.8	4.0
	3-xor comb	6.5	6.3	5.3
	2-xor, 2 reg	6.8	6.3	5.9
	3-xor, 2 reg	6.7	7.5	6.3
Parallel Vectors (4 config.)	Illinois	458	470	466
	2-xor comb	403	412	465
	3-xor comb	472	512	588
	2-xor, 2 reg	451	510	490
	3-xor, 2 reg	465	472	590
Serial Vectors (4 config.)	Illinois	13	18	42
	2-xor comb	3	12	18
	3-xor comb	0	2	6
	2-xor, 2 reg	0	2	7
	3-xor, 2 reg	0	0	2
ATPG Runtime	Illinois	3.14	3.26	3.44
	2-xor comb	5.40	3.20	5.24
	3-xor comb	6.65	7.20	7.24
	2-xor, 2 reg	5.20	4.76	6.25
	3-xor, 2 reg	7.10	7.14	7.24

Table 6.3: Results for Design A

	Num. Scan Chains			
	Decompressor	64	128	192
Compression With 1 config.	Illinois	2.9	4.4	4.6
	2-xor comb	3.7	4.7	4.7
	3-xor comb	4.1	5.5	6.0
	2-xor, 1 reg	4.8	5.5	6.2
	2-xor, 2 reg	4.8	5.9	7.2
	3-xor, 1-reg	4.9	5.7	6.4
	3-xor, 2-reg	4.9	5.7	7.6
Parallel Vectors (1 config.)	Illinois	791	810	853
	2-xor comb	776	785	780
	3-xor comb	795	814	830
	2-xor, 1 reg	760	821	787
	2-xor, 2 reg	795	797	883
	3-xor, 1 reg	798	824	807
	3-xor, 2 reg	775	793	804
Serial Vectors (1 config.)	Illinois	168	132	137
	2-xor comb	118	120	138
	3-xor comb	95	93	98
	2-xor, 1 reg	75	92	96
	2-xor, 2 reg	66	74	72
	3-xor, 1 reg	64	88	90
	3-xor, 2 reg	66	89	70
Coverage with parallel vectors only (1 config.)	Illinois	92.1	92.5	90.2
	2-xor comb	93.4	93.5	93.2
	3-xor comb	93.2	93.1	92.9
	2-xor, 1 reg	92.2	92.8	93.4
	2-xor, 2 reg	94.1	93.9	94.2
	3-xor, 1 reg	94.3	94.0	94.6
	3-xor, 2 reg	95.3	97.8	97.6
Compression with 4 configs.	Illinois	3.6	4.5	4.8
	2-xor comb	4.0	4.9	5.4
	3-xor comb	4.8	6.1	6.0
	2-xor, 1 reg	4.8	6.5	7.8
	2-xor, 2 reg	4.8	6.2	7.9
	3-xor, 1 reg	4.7	6.6	7.8
	3-xor, 2 reg	5.5	6.6	8.0
Parallel Vectors (4 config.)	Illinois	807	902	1102
	2-xor comb	798	822	878
	3-xor comb	885	920	995
	2-xor, 1 reg	1133	1512	1588
	2-xor, 2 reg	1139	1556	1498
	3-xor, 1 reg	1188	1590	1588
	3-xor, 2 reg	1002	1603	1616
Serial Vectors	Illinois	121	122	119
	2-xor comb	98	110	112

(4 config.)	3-xor comb	55	73	90
	2-xor, 1 reg	24	27	36
	2-xor, 2 reg	23	30	38
	3-xor, 1 reg	22	21	36
	3-xor, 2 reg	18	20	32
ATPG Runtime (sec)	Illinois	152	148	149
	2-xor comb	155	149	155
	3-xor comb	162	166	164
	2-xor, 1 reg	160	158	150
	2-xor, 2 reg	158	166	171
	3-xor, 1 reg	162	162	159
	3-xor, 2 reg	169	166	169

Table 6.4: Results for Design B

	Num. Scan Chains			
	Decompressor	64	128	192
Compression (1 config.)	Illinois	1.2	1.3	1.2
	2-xor comb	1.6	1.6	1.5
	3-xor comb	1.5	1.8	1.5
	2-xor, 1 reg	1.8	1.8	1.7
	2-xor, 2 reg	1.8	1.9	1.8
	3-xor, 1 reg	1.9	2.0	2.0
	3-xor, 2 reg	2.4	3.2	2.5
Parallel Vectors (1 config.)	Illinois	106	88	108
	2-xor comb	135	164	122
	3-xor comb	190	189	202
	2-xor, 1 reg	199	238	192
	2-xor, 2 reg	248	294	283
	3-xor, 1 reg	243	287	347
	3-xor, 2 reg	252	330	313
Serial Vectors (1 config.)	Illinois	99	105	118
	2-xor comb	74	80	93
	3-xor comb	74	72	93
	2-xor, 1 reg	50	66	80
	2-xor, 2 reg	54	64	72
	3-xor, 1 reg	50	55	60
	3-xor, 2 reg	36	45	48
Coverage with parallel vectors only (1 config.)	Illinois	91.1	89.6	89.7
	2-xor comb	94.1	92.3	91.1
	3-xor comb	95.3	94.6	93.5
	2-xor, 1 reg	96.6	95.8	93.8
	2-xor, 2 reg	97.2	96.6	96.7
	3-xor, 1 reg	97.8	97.2	96.8
	3-xor, 2 reg	98.2	98.2	97.6
Compression (4 configs.)	Illinois	1.4	1.5	1.3
	2-xor comb	1.7	2.1	1.9
	3-xor comb	1.9	2.3	1.8
	2-xor, 1 reg	2.1	2.1	2.0
	2-xor, 2 reg	2.1	2.6	2.5
	3-xor, 1 reg	2.2	2.6	2.4
	3-xor, 2 reg	2.5	3.3	3.1
Parallel Vectors (4 config.)	Illinois	342	346	366
	2-xor comb	270	351	362
	3-xor comb	270	380	356
	2-xor, 1 reg	309	386	344
	2-xor, 2 reg	301	387	361
	3-xor, 1 reg	299	383	456
	3-xor, 2 reg	269	371	427
	Illinois	66	75	98
	2-xor comb	54	50	63

Serial Vectors (4 config.)	3-xor comb	44	42	63
	2-xor, 1 reg	30	46	60
	2-xor, 2 reg	34	34	45
	3-xor, 1 reg	30	35	42
	3-xor, 2 reg	28	27	30
ATPG Runtime (sec)	Illinois	0.97	0.96	0.98
	2-xor comb	1.08	1.26	1.92
	3-xor comb	2.12	2.22	2.80
	2-xor, 1 reg	1.10	1.87	2.57
	2-xor, 2 reg	1.70	2.14	2.36
	3-xor, 1 reg	2.62	2.81	3.26
	3-xor, 2 reg	2.31	2.76	2.98

6.7 CONCLUSIONS

The results in this chapter show that by using limited depth sequential decompression, a significant improvement in compression can be achieved. A number of commercial test compression schemes are based on incorporating the decompressor constraints in the ATPG search/backtrace. The proposed method provides a simple and practical way to boost the effectiveness of such schemes by incorporating tester slice registers to allow the use of free-variables across multiple clock cycles.

One area for future research would be to investigate how structural information about the logic cones in the circuit-under-test could be used to improve the design of limited depth sequential decompressors.

Chapter 7: Conclusion and Future Work

7.1 CONCLUSION

As mentioned in Chapter 1, circuit reliability has become an important design consideration. This dissertation proposes several concurrent error detection/correction methodologies to address the problems arising from the different threats to circuit reliability. The proposed techniques primarily target soft errors that occur randomly and depend on factors like alpha-particles or gamma-radiation. As process technology scales well below 100 nanometers, the higher operating frequencies, lower voltage levels, and smaller noise margins make integrated circuits increasingly susceptible to SEUs resulting in a dramatic increase in soft errors. Due to their irregular structures, concurrent error detection in combinational logic circuits is difficult. In chapter 2, a non-intrusive concurrent error detection technique is presented. The advantage of the proposed scheme is the easy trade-off between error coverage and area overhead. It has been shown that the most likely errors can be detected using a fraction of the overhead compared to duplication. The problem of soft error is even more prominent in memories. However it is easier to employ error correction schemes in memories due to their regular structures. In chapter 2, a low cost error correcting code is proposed to design multiple bit upset tolerant memories. The code is designed by a heuristic search algorithm and codes for different word sizes are provided. The proposed codes have the least overhead for the targeted application amongst all the known codes. In chapter 4, the search algorithm was extended to derive an unequal error protection code with even lesser overhead. These codes are very useful to protect data in router memories. These codes provide different levels of error protection for the different portions of the packet. In chapter 5, a runtime

and memory efficient algorithm was presented to accurately estimate signal probabilities of the circuit lines. This algorithm can be used to estimate soft error susceptibility of different nodes in the circuit. The estimation of soft error susceptibility helps in insertion of proper error protection schemes in the circuit. Finally, some problems in the area of off-line testing were looked at. In chapter 6, a technique was proposed to reduce the deterministic test data volume and test time using a limited depth sequential expansion strategy.

7.2 FUTURE WORK

This dissertation opens different alleys for future research. The search based code design strategy can be used for designing codes targeting a specific application. A different set of constraints have to be implemented during the search to achieve the desired code. For example, the proposed code design strategy can be used to design codes for multilevel memories. By accurately designing the constraints from the error conditions accurately, a low cost code can be designed to protect data in multi-level memories. The proposed codes can also be extended to higher order Galois fields. The proposed scheme for deriving unequal protection code can be used to derive codes that provide higher protection for certain bits (may not be contiguous). Another approach could be to partition the code space into mutually exclusive groups of codewords. The error correction capacity for each group could be varied as per the targeted application. A possible application for these kinds of codes is in the wormhole routing used in network on chip or in communication applications. Unlike the store and forward routing, in the wormhole routing the packets are split into smaller sized packets called flits. This kind of code can provide different levels of protection for the different flits. For example the header and the trailer flits can be given more protection compared to the data flits. The scheme used to derive the SEC-DED-DAEC code can be extended to correct any subset

of double-bit errors not necessarily adjacent. The test data compression technique can further be improved by incorporating the circuit information while constructing the decompressor. The proposed design of the decompressor does not take into account any circuit information and hence can be designed independent of the circuit under test. However the linear dependencies among the decompressor outputs can be reduced by using the knowledge about the structure of the circuit. This dissertation primarily focuses on the concurrent error detection / correction methodologies for memories and combinational logic circuits. A natural extension of the work is to look into reliability issues in the emerging nano-technology. The existing concurrent error detection/correction methodologies may not be directly applicable for these emerging technologies. Future work might involve adapting the existing CED schemes for the emerging technologies.

Bibliography

[Abramson 59] Abramson N. M., ”A Class of Systematic Codes for Non-Independent Errors”, Proc. IRE Trans. on Information Theory, Vol. IT-5, pp. 150-157, Dec. 1959.

[Agrawal 78] Agrawal, V. D., “When to Use Random Testing,” IEEE Transactions on Computers, Vol C-27, No. 11, November 1978, pp. 1054-1055.

[Akers 78] Akers, S. B. “Binary Decision Diagrams,” IEEE Transactions on Computers. Vol c-27, No. 5, May 1978, pp 509-516.

[Alexandrescu 02] Alexandrescu, D., L. Anghel, and M. Niholaidis, “New Methods for Evaluating the Impact of Single Event Transients in VDSM ICs,” Prof. of Int. Symp. on Defect and Fault Tolerance, pp. 99-107, 2002.

[Al-Kharji 97] Al-Kharji, Masaed., Sami A. Al-Arian,”Anew Heuristic Algorithm for Estimating Signal and Detection Probabilities”, 7th Great Lake Symposium on VLSI, p. 26, 1997.

[Almukhaizim 04a] S. Almukhaizim, P. Drineas, and Y. Makris, "Concurrent Error Detection for Combinational and Sequential Logic via Output Compaction," Proc. of Int. Symp. on Quality Electronic Design, pp. 319-324, 2004.

[Almukhaizim 04b] S. Almukhaizim, P. Drineas, and Y. Makris, "Cost-Driven Selection of Parity Trees," Proc. of VLSI Test Symposium, pp. 319-324, 2004.

[Aloul 02] Aloul, Fadi A., Igor. L. Markov and Karem. A. Sakallah,”Improving the Efficiency of Circuit-to-BDD Conversion by Gate and Input Ordering”, Int. Conf. On Computer Design, 2002.

[Bayraktaroglu 03] I. Bayraktaroglu and A. Orailoglu, “Concurrent Application of Compaction and Compression for Test Time and Data Volume Reduction in Scan Designs,” IEEE Trans. on Computers, Vol. 52, No. 11, pp. 1480-1489, Nov. 2003.

[Berlekamp 68] Berlekamp, E. R., Algebraic Coding Theory, McGraw-Hill, New York, 1968.

[Bernstein 63] Bernstein, A., and W. Kim,”Single and Double Adjacent Error-correcting codes for arithmetic Units”, IEEE Tran. on Information Theory, Vol. 9, pp. 209-210, Mar. 1963.

[Bertozzi 02] Bertozzi, D. L. Benini, G. De Micheli, "Low power error resilient encoding for on-chip data buses", *Proc. of Design, Automation and Test in Europe Conference*, pp. 102-109, 2002.

[Bodnar 03] Bodnar, L. and G. Chapelle,"A single error correctiondouble burst error detection code", *Proc. of Asilomar Conference on Signals, Systems and Computers*, Vol. 1, pp. 1118-1121, Nov, 2003.

[Bolchini 97] C. Bolchini, F. Salice, and D. Sciuto, "A Novel Methodology for Designing TSC Networks based on the Parity Bit Code," Proc. European Design and Test Conference, pp. 440-444, 1997.

[Bossen 70] Bossen, D. C.,"b-Adjacent Error Correction", IBM Journal of Research and Development, Vol. 14, pp. 402-408, Jul. 1970.

[Bowman 03] Bowman, R.C., "Technology scaling trends and accelerated testing for soft errors in commercial silicon devices", Proc. IEEE International On-Line Testing Symposium, pp. 4, 2003.

[Bowman 04] Bowman, R.C., "Soft errors in commercial integrated circuits", International Journal of High Speed Computing, Vol. 14, No.2, pp. 299-309, 2004.

[Brglez 85] Brglez, F., H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in FORTRAN," Proc. IEEE Symp. on Circ. Syst. (ISCAS), Kyoto, Japan, June 1985, pp. 695-698.

[Brglez 84] Brglez, F., "On Testability Analysis of Combinational Networks," in Proc. IEEE Symp on Circ. Syst., 1984, pp. 221-225.

[Bryant 86] Bryant, R. E. "Graph Based Algorithms for Boolean Function Manipulation," IEEE Transactions on Computers, Vol c-35, No. 8, August 1986, pp. 677-691.

[Butler 91] Butler, K. M., D. E. Ross, R. Kapur, and M. R. Mercer, "Heuristics to Compute Variable Orderings for Efficient Manipulation of OBDDs," Proceeclings of the 28th ACR.I/lEEE Design Aritomation Conference, San Francisco, California, June 17-19, 1991, pp. 417-420.

[Calin 95] Calin, Th., F. L. Vargas, and M. Nicolaidis,"Upset Tolerant CMOS Using Current Monitoring: Prototype and Test Experiments", Proc. Int. Test Conference, pp. 45-53, 1995.

[Chakravarty 90] Chakravarty, S., and H. Hunt III, "On Computing Signal Probability and Detection Probability of Stuck-at Faults," IEEE Transactions on Computers, Vol. 39. No. 11. Nov. 1990.

[Chen 68] Chen, C. L.," Error Correcting Codes with Byte Error Detection Capability", IEEE Trans. On Computers, Vol. C-32, pp. 615-621, May 1983.

[Chen 96] Chen, C. L.,"Symbol Error Correcting Codes for Memory Applications", Proc. of Fault Tolerant Computing Systems, pp. 200-207, 1996.

[Cohen 99] N. Cohen, et al., "Soft Error Considerations for Deep-Submicron CMOS Circuit Applications," International Electron Devices Meeting, 1999.

[Costa 97] Costa, Jose., Jose C. Monteiro and Srinivas Devadas,"Switching Activity Estimation Using Limited Depth Reconvergent Path Analysis," in Proc. of International Symposium on Low Power Electronics and Design, PP. 184-189, 1997.

[Das 99] D. Das and N. A. Touba, "Synthesis of Circuits with Low-Cost Concurrent Error Detection Based on Bose-Lin Codes," Journal of Electronic Testing: Theory and Applications, Vol. 15, Nos. 1/2, pp. 145-155, Aug. 1999.

[De 94] K. De, et al., "RSYN: A System for Automated Synthesis of Reliable Multilevel Circuits," IEEE Trans. VLSI Systems, pp. 186-195, Jun. 1994.

[Elspas 60] Elspas, B.,"A Note on p-nary Adjacent-error-correcting Codes", IEEE Trans. on Information Theory, Vol. 6, Mar. 1960.

[Favalli 02] M. Favalli and C. Metra, "Online Testing Approach for Very Deep-Submicron ICs," IEEE Design and Test of Computers, Vol. 19, No. 2, pp. 16-23, Mar. 2002.

[Franco 94] P. Franco and E. J. McCluskey, "On Line Delay Testing of Digital Circuits," Proc. VLSI Test Symposium, pp. 167-173, 1994.

[Frantz 06] Frantz, A. P., F.L. Kastensmidt, L. Carro, and E. Cota, "Exploiting ECC redundancy to minimize crosstalk impact", *Procs. of annual symposium on integrated circuits and systems design*, pp. 202-207, 2006.

[Fujiwara 87] Fujiwara, E., and K. Matsuoka, "A Self-Checking Generalized Prediction Checker and Its Use for Built-In Testing," IEEE Trans. Computers, Vol. C-36, No. 1, pp. 86-93, Jan. 1987.

[Fuziwara 98] Fujiwara, E., T. Ritthongpitak and M. Kitami,"Optimal Two-Level Unequal Error Control Codes for Computer Systems", *IEEE Trans. on Computers*, Vol. 47, No. 12, pp. 1313- 1325, Dec. 1998.

[Gill 05] Gill, B., M. Nicolaidis, and C. Papachristou," Radiation Induced Single-Word Multiple-bit Upsets Correction in SRAM", Proc. of Int. Online Test Symposium, pp. 266-271, Jul. 2005.

[Gössel 93] M. Gössel and S. Graf, Error Detection Circuits, McGraw-Hill Book Company, London, 1993.

[Hamming 50] Hamming, R.W., ”Error Correcting and Error Detecting Codes”, Bell Sys. Tech. Journal, Vol. 29, pp. 147-160, Apr. 1950.

[Hamzaoglu 99] I.Hamzaoglu and J.H.Patel, “Reducing Test Application Time for Full Scan Embedded Cores,” Proc. Int. Symp. on Fault-Tolerant Computing, pp. 260-267, 1999.

[Hayashi 00] Hayashu, T., and E. Fujiwara,”Bit and Byte Error Protection Codes with Two Protection Levels,” *Trans. IEICE A*, Vol. J83-A, No. 2, pp. 196-207, Feb. 2000.

[Hsiao 70] Hsiao, M. Y., ”A Class of Optimal Minimum Odd-weight-column SEC-DED codes”, IBM Journal of Research and Development, Vol. 14, pp. 395-401, 1970.

[Jain 85] Jain S.K., and V.D. Agrawal, “Statistical Fault Analysis,” IEEE Design & Test of Computers, pp. 38-45, 1985.

[Jha 93] N. K. Jha and S. Wang, “Design and Synthesis of Self-Checking VLSI Circuits,” IEEE Trans. Computer-Aided Design, Vol. 12, No. 6, pp. 878-887, Jun. 1993.

[Kastensmidt 06] Kastensmidt, F., L. Carro, R. Reis, *Fault-Tolerance Techniques for SRAM-based FPGAs*, Series: Frontiers in Electronic Testing, Springer, Vol. 32, pp. 180-185, 2006.

[Kawakami 04] Kawakami, Y., et al., ”Investigation of Soft Error Rate Including Multi-Bit Upsets in Advanced SRAM Using Neutron Irradiation Test and 3D Mixed-mode Device Simulation”, Proc. of IEEE Int’l Electronic Device Meeting, pp. 945-948, Dec. 2004.

[Kodavarti 93] Kodavarti, R., and D. Ross,” Signal Probability Calculations Using Partial Functional Manipulations,” in Proceedings of Eleventh Annual 1993 IEEE VLSI Test Symposium, pp. 194-200,April 1993.

[Könemann 01] B. Koenemann, C. Barnhart, B. Keller, T. Snethen, O. Farnsworth, and D. Wheater, “A SmartBIST Variant with Guaranteed Encoding,” Proc. Asian Test Symp., pp. 325-330, 2001.

[Krewell 05] Krewell.,”Multicore Showdown”, *Microprocessor Report,* vol. 19, pp. 41-45,2005.

[Krishna 01] C.V. Krishna, A. Jas, and N.A. Touba, “Test Vector Encoding Using Partial LFSR Reseeding,” Proc. Int. Test Conf., pp. 885-893, 2001.

[Krishna 04] C.V. Krishna, N.A. Touba, "3-Stage Variable Length Continuous-Flow Scan Vector Decompression Scheme," Proc. of VLSI Test Conf., pp. 79-86, 2004.

[Lajolo 01] Lajolo, M.; "Bus guardians: an effective solution for online detection and correction of faults affecting system-on-chip buses", *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, Vol. 9, Iss. 6, pp. 974–982, 2001.

[Lala 78] Lala, P. K.," An Adaptive Double Error Correction Scheme for Semiconductor Memory Systems," Digital Processes, Vol. 4, pp. 237-243, 1978.

[Lee 59] Lee, Y.,"Representation of Switching Circuits in Binary-decision Programs," Bell Syst. Tech. J., Vol 38. July 1959, pp. 985-999.

[Lee 98] K.-J. Lee, J.J. Chen, and C.H. Huang, "Using a Single Input to Support Multiple Scan Chains," Proc. Int. Conf. on Computer-Aided Design, pp. 74-78, 1998.

[Lin 83] Lin, S., and D. J. Costello, Jr., Error Control Coding: Fundamentals and Applications, Prentice-Hall, 1983.

[Maiz 03] Maiz, J., S. Hareland, K. Zhang, and P. Armstrong, "Characterization of Multibit Soft Error Events in Advanced SRAMs", Proc. of IEEE Int'l Electronic Device Meeting, pp. 519-522, Dec. 2003.

[Makihara 00] Makihara, A., et al., "Analysis of Single-Ion Multiple-Bit Upset in High-Density DRAMS", IEEE Trans. on Nuclear Science, Vol. 47, No. 6, Dec. 2000.

[Markowsky 87] Markowsky, G., "Bounding Signal probabilities in Combinational Circuits," IEEE Transactions on Computers, Vol c-36, No. 10, October 1987, pp. 1247-1251.

[Masnick 67] Masnick, B., and J.K. Wolf, "On Linear Unequal Error Protection Codes," *IEEE Trans. Inf. Theory*, Vol. IT-13, No. 4, pp. 600-607, Oct. 1967.

[McCluskey 88] McCluskey, E.J., S. Makar, S. Mourad, and K. Wagner, "Probability Models for Pseudorandom Test Sequences," IEEE Trans. on Computer-Aided Design, Vol. 7, No. 1, pp. 68-74, Jan. 1988.

[Mercer 92] Mercer, M. R., R. Kapur and D.E. Ross, "Functional Approaches to Generate Orderings for Efficient Synibolic Representations," Proc. ACMIEEE 29th Design Automation Conf., .June 1992.

[Metra 98] C. Metra, M. Favalli, and B. Ricco, "Online Detection of Logic Errors Due to Crosstalk, Delay and Transient Faults," Proc. International Test Conference, pp. 524-533, 1998.

[Mitra 06] S. Mitra and K.S. Kim, "XPAND: An Efficient Test Stimulus Compression Technique," IEEE Trans. on Computers, Vol. 55, No. 2, pp. 163-173, Feb. 2006.

[Mohanram 03a] Mohanram, K., E.S. Sogomonyan, M. Gössel, and N.A. Touba, "Synthesis of Low-Cost Parity-Based Partially Self-Checking Circuits," Proc. of International On-Line Test Symposium, pp. 35-40, 2003.

[Mohanram 03b] Mohanram, K., and N.A. Touba, "Cost-Effective Approach for Reducing Soft Error Failure Rate in Logic Circuits," Proc. of International Test Conference, pp. 893-901, 2003.

[Morelos-Zaragoza 94] Morelos-Zaragoza, R.H., and S.Lin," On a Class of Optimal Nonbinary Linear Unequal-Error-Protection Codes for Two Sets of Messages,"*IEEE Trans. Inf. Theory*, Vol. IT-40, No. 1, pp. 196-200, Jan. 1994.

[Morozov 00] Morozov, A., V.V. Saposhnikov, Vl.V. Saposhnikov, and M. Gössel, "New Self-Checking Circuits by Use of Berger-Codes," Proc. of On-Line Testing Workshop, 2000, pp. 141-146, 2000.

[Mrugalski 04] G. Mruglaski, J. Rajski, and J. Tyszer, "Ring Generators – New Devices for Embedded Test Applications," IEEE Trans. on Computer-Aided Design, Vol. 23, No. 9, pp. 1306-1320, Sept. 2004.

[Namba 01] Namba, K., and E. Fujiwara,"Unequal Error Protection Codes with Two-Level Burst and Bit Error Correcting Capabilitie", *IEEE International Symposium on Defect and Fault Tolerances*, Vol. 47, No. 12, pp. 1313- 1325, Dec. 1998.

[Nicolaidis 98] M. Nicolaidis and Y. Zorian, "Online Testing for VLSI – A Compendium of Approaches," Journal of Electronic Testing: Theory and Applications, Vol. 12, Nos. 1/2, pp. 7-20, Feb. 1998.

[Nicolaidis 99] M. Nicolaidis, "Time Redundancy Based Soft-Error Tolerance to Rescue Nanometer Technologies," Proc. of VLSI Test Symposium, pp. 86-94, 1999.

[Nicolaidis 05] Nicolaidis, M, "Design for soft error mitigation**",** *IEEE Trans. on Device and Materials Reliability*, Vol. 5, Iss. 3, pp. 405-418, 2005.

[Nieuwland 05] Nieuwland, A.K.; A. Katoch, D. Rossi, C. Metra, "Coding techniques for low switching noise in fault tolerant busses", *Proc. of International On-Line Testing Symposium*, pp. 183-189, 6-8 July 2005.

[Pandey 02b] A.R. Pandey and J.H. Patel, "Reconfiguration Technique for Reducing Test Time and Test Volume in Illinois Scan Architecture Based Designs," Proc. VLSI Test Symposium, pp. 9-15, 2002.

[Parker 75] Parker, K. P., and E. J. McClusky, "Probabilistic Treatment of General Combinational Networks," Transactions on Computers, pp. 668-670, 1975.

[Park 06] Park, D., C. Nicopoulos, J. Kim, N. Vijaykrishnan, and C.R. Das, " Exploring Fault-Tolerant Network-on-Chip Architectures", *Procs. of International Conference on Dependable systems and Networks (DSN 06)*, pp. 93-104, 2006.

[Rajski 04] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherje, "Embedded Deterministic Test," IEEE Trans. on Computer-Aided Design, Vol. 23, No. 5, pp. 776-792, May 2004.

[Ravi 98] Ravi, Kavita., Kenneth L. McMillan, Thomas R. Shiple and Fabio Somenzi., "Approximation and Decomposition of Binary Decision Diagrams," Proc of Design Automation Conference, 1998.

[Reddy 78] Reddy, S.M., "A Class of Linear Codes for Error Control in Byte-per-Package Organized Memory Systems", IEEE Trans. On Computers, Vol. C-27, pp. 455-458, May. 1978.

[Reed 60] Reed, I. S., and G. Solomon,"Polynomial Codes Over Certain Fields",J. Soc. Ind. Appl. Mat., Vol. 8, pp. 300-304, Jun. 1960.

[Rejimon 05] Rejimon, Thara., and Sanjukta Bhanja,"An Accurate Probabilistic Model for Error Detection," 18th International Conference on VLSI Design, pp717-722, 2005.

[Ross 90] Ross, D. E., "Functional Calculation Using Ordered Partial Multi Decision Diagrams," Ph. D. dissertation. University of Texas, Austin, August 1990.

[Ross 91] Ross, D. E., K. M. Butler, R. Kapur, and M. R. Mercer, "Fast Functional Evaluation of Candidate OBDD Variable Orderings," Proc. of The European Conference on Design Automation, Amsterdam, 'The Netherlands, February 25-28, 1991, pp. 11-10.

[Rossi 05] Rossi, D., C. Metra, A.K. Nieuwland, and A. Katoch, "Exploiting ECC redundancy to minimize crosstalk impact", *IEEE Design and Test of Computers*, Vol. 22, Iss. 1, pp. 59-70, Jan 2005.

[Samaranayake 03] S. Samaranayake, E. Gizdarski, N. Sitchinava, F. Neuveux, R. Kapur, and T. W. Williams, "A Reconfigurable Shared Scan-In Architecture," Proc. VLSI Test Symposium, pp. 9-14, 2003.

[Saposhnikov 96] Saposhnikov, Vl.V., A. Dmitriev, M. Gössel, and V.V. Saposhnikov, "Self-Dual Parity Checking – A New Method On-Line Testing," Proc. of VLSI Test Symposium, pp. 162-168, 1996.

[Saposhnikov 98a] Saposhnikov, Vl.V., V.V. Saposhnikov, A. Dmitriev, and M. Gössel, "Self-Dual Duplication for Error Detection," Proc. of Asian Test Symp., pp. 296-300, 1998.

[Saposhnikov 98b] V. V. Saposhnikov, et al., "A New Design Methodology for Self-Checking Unidirectional Combinational Circuits," Journal on Electronic Testing: Theory and Applications, Vol. 12, Nos. 1/2, pp. 41-53, Feb. 1998.

[Satoh 00] Satoh, S., Y. Tosaka, S.A. Wender, "Geometric Effect of Multiple-bit Soft Errors Induced by Cosmic-ray Neutrons on DRAMs", Proc. of IEEE Int'l Electronic Device Meeting, pp. 310-312, Jun. 2000.

[Savir 80] Savir, J., G. S. Ditlow, and P. H. Bardell, "Random Pattern Testability," IEEE Transactions on Computers,Vol c-33, No. 1, January 1984, pp. 79-90.

[Savir 90] Savir, J. "Improved Cutting Algorithm," IBM Journal of Research and Development, Vol 34, No. 2/3, March/May 1990, pp. 40-75.

[Seth 85] Seth, C., L. Pan and V. D. Agrawal, "PREDICT- Probabilistic Estimation of Digital Circuit Testability," Fault-Tolerant Computing Symposium, Ann-Arbor, MI, pp.220-225, June 1985.

[Seth 89] Seth, C., V. D. Agrawal, "A New Model for Computation of Probabilistic Observability," Integration, the VLSI Journal, Volume 7, 1989 pp.49-75.

[Shivakumar 02] Shivakumar, P., M. Kistler, S.W. Keckler, D. Burger, and L. Alvisi, "Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic," Proc. International Conference on Dependable Systems and Networks, pp. 389-398, 2002.

[Sitchinava 04] N. Sitchinava, S. Samaranayake, R. Kapur, E. Gizdarski, F. Neuveux, and T.W. Williams, "Changing the Scan Enable During Shift," Proc. VLSI Test Symposium, pp. 73-78, 2004.

[Sogomonyan 74] Sogomonvan, E., "Design of Built-In Self-Checking Monitoring Circuits for Combinational Devices," Automation and Remote Control, Vol. 35, No. 2, pp. 280-289, 1974.

[Tang 03] H. Tang, S.M. Reddy, and I. Pomeranz, "On Reducing Test Data Volume and Test Application Time for Multiple Scan Designs," *Proc. Int. Test Conf.,* pp. 1079-1088, 2003.

[Tarnick 94] Tarnick, S., "Bounding Error Masking in Linear Output Space Compression Schemes," Proc. of Asian Test Symposium, pp. 27-32, 1994.

[Touba 97] N. A. Touba and E. J. McCluskey, "Logic Synthesis of Multilevel Circuits with Concurrent Error Detection," IEEE Trans. on Computer-Aided Design, Vol. 16, No. 7, pp. 783-789, Jul. 1997.

[Uchino 95] Uchino, Taku., Fumihiro Minami, Takashi Mitsuhashi and Nobuyuki Goto," Switching Activity Analysis using Boolean Approximation Method," International Conference on Computer-Aided Design,pp.20-25, 1995.

[Vargas 94] Vargas, F. L., and M. Nicolaidis," SEU-Tolerant SRAM Design Based On Current Monitoring", Proc. Int. Symposium pn Fault Tolerant Computing, pp. 106-115, June 1994.

[Wang 04] L.-T. Wang, X. Wen, H. Furukawa, F.-S. Hsu, S.-H. Lin, S.-W. Tsai, K. S. Abdel-Hafez, and S. Wu, "VirtualScan: A New Compressed Scan Technology for Test Cost Reduction," Proc. Int. Test Conf., pp. 916-925, 2004.

[Wolf 69] Wolf, J. K.," Adding Two Information Symbols to Certain Non-Binary BCH Codes and Some Applications", Bell Systems Technical Journal, Vol. 48, pp. 2405-2424, 1969.

[Wunderlich 85] Wunderlich, J., "PROTEST: A Tool for Probabilistic Analysis," in Proc. 22nd Design Automation Conf., Las Vegas, June 23-25, 1985. pp. 204-211.

[Yang 91] S. Yang, "Logic Synthesis and Optimization benchmarks,Version 3.0, Tech. Report, Microelectronics Centre of North Carolina, 1991.

[Ziegler 96] J. F. Ziegler, et al., "IBM Experiments in Soft Fails in Computer Electronics (1978-1994)," IBM Journal of Research and Development, Vol. 40, pp. 3-18, 1996.

Vita

Avijit Dutta was born and raised in West Bengal, India. He did his undergraduate work from Jadavpur University, West Bengal, India, where he majored in Computer Science and Engineering with Honors in the year 2000. He worked in Cadence Design Systems from 2000 to 2003. He joined the University of Texas at Austin in 2003, where he did his MS in Computer Engineering with a thesis on VLSI Testing in the year 2005. He is currently doing his PhD in Computer Engineering with a focus on synthesis for circuit reliability and test data compression. His primary areas of research interests include synthesis for circuit reliability, fault tolerant computing, coding theory and test data compression / decompression techniques.

Permanent Address : 58/26A, Prince Anwar Shah Road
Lake Gardens, Calcutta 700045
West Bengal, India

This dissertation was typed by Avijit Dutta.